THE COMPUTER PROJECTS MAGAZINE MAY 1984

Free Inside
YOUR ROBOT
Win a Robot Competition

ELECTRONICS & COMPUTING MONTHLY

AN EMAP PUBLICATION

USA $2.95
Germany D5.80 85p

FORGET YOUR MEMORY PROBLEMS

BUILD EXTRA K INTO YOUR BBC

EXCLUSIVE
Amstrad's computer full technical specs

GRAPHICS-THE THIRD DIMENSION

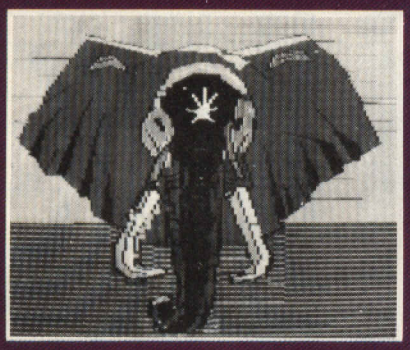
68008-BREAK INTO QL MACHINE CODE

MODEMS SURVEYED • CBM 64 INTERFACE

ELECTRONICS & COMPUTING

Contents

Vol. 4 Issue 5



COVER PHOTO BY ROB BRIMSON

Electronics & Computing Monthly
Scriptor Court, 155 Farringdon Road,
London, EC1R 3AD

Editorial 01-833-0846
Editor Gary Evans
Assistant Editor William Owen
Production Editor Liz Gregory
Administration Serena Hadley

Advertising 01-833-0531
Chief Executive Richard Jansz
Classified Martin Derx

Production 01-833-0531
Art Editor Jeremy Webb
Make-up Time Graphics
Publisher Alfred Rolington

Distribution
EMAP National Publications

Published by
EMAP Business and Computer Publications

Printed by
Riverside Press, England

Subscriptions
Electronics & Computing Monthly, (Subscription Department), Competition House, Farndon Road, Market Harborough, Leicestershire.

Electronics & Computing Monthly is normally published on the 13th day of each month.

ABC

PROJECTS

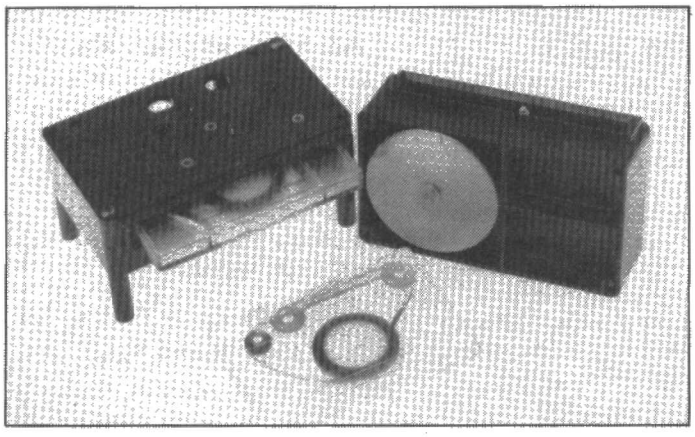
FEATURES

REVIEWS

PLUS

33

NEWS NEWS NEWS NEWS



New storage concepts

The stringy floppy is an increasingly popular form of storage in the home computer market. Quicker than cassette tapes and cheaper than disks, there are now at least three such devices on the market.

The first is of course Sinclair's Microdrive, used with the Spectrum and the QL. Ikon Computer Products market the Ultra-Drive, which is available for the Dragon, with versions for the BBC, Nascom, Tandy TRS80, Oric, Electron and Commodore 64 to follow at monthly intervals from April.

The Ultra-Drive has a read/write speed of 1200 characters per second (about 10 times faster than a conventional cassette recorder) and a capacity of 200K per cassette. The device costs £79.95 plus £3.45 p&p.

The Astec Stringy Floppy has a smaller memory capacity than the Ultra-Drive but stores data at just over twice the speed (2625 characters per second) and can be used with any micro with an RS232 interface.

Data is stored in a disc-like block structure to allow maximum utilisation of the tape. Since the data format on the tape is standard, data transmission across different systems will be assured even if the interfaces run at different speeds.

Ikon Computer Products, Kiln Lane, Laugharne, Carmarthen, Dyfed SA33 4QE. Astec, 0734 509411.

Enterprise flans Elan

Enterprise Computers, after much agony, are now ready to baptise their baby but may have missed the boat.

The computer was 'launched' at the Personal Computer World Show in September as the Elan, but following action by Elan Digital Systems of Crawley, a new monicker, the Flan, was given to the computer. Enterprise (sensibly) now claim this was a joke and have settled finally for 'Enterprise'.

Not quite enough enterprise is evident. The computer will not be available for sale until a full year after the original announcement. The 64K £200 machine might have been successful if it had arrived before the QL, Amstrad and Electron, but by next September its price and performance will doubtless be surpassed.

Chip consumers starve

In recent weeks, standard 74xx series ICs have become scarcer than the Sinclair QL. Even humble devices such as the 7400 quad NAND gate are difficult to locate and, if you can find some, the price quoted can be anything up to 10 times the one-off price quoted only a couple of months back.

Nazir Jessa of Watford Electronics confirmed that there was a definite famine but went on to say that Watford were looking after the one-off customer. In a complete reversal of normal pricing policy Nazir Jessa is charging customers ordering in quantities of several thousand prices that are in some cases double the one-off level – and they are happy to pay.

Recent bulk shipments from Watford have gone as far afield as mainland Europe and, in a distinctly coals to Newcastle fashion, to Singapore. The spot prices commanded by some devices is causing many manufacturers, among them Acorn, more than a few problems.

The conflicting requirement of maintaining production while keeping costs within budget are becoming increasingly difficult to satisfy. The famine is so acute that orders of devices in the 25,000 bracket are being met with deliveries of perhaps 200 or so a month.

The reasons for the current famine are two fold. The first is that, a couple of years ago, the major manufacturers over-estimated the demand for TTL ICs and flooded the market with devices. The glut caused prices to fall to the levels that have been prevelent in recent times. Having had their fingers burned in the past, this time around manufacturers have been reluctant to commit resources to the production of this family of devices. The second reason for the shortage is the rapid expansion in the personal computer market. The TTL family features greatly in the design of machines such as the Commodore 64 and the BBC micro.

Manufacturers of this sort of product have cornered the market in some of the devices that they use in quantity. There is an element of hoarding here: some manufacturers stock up devices for which they have no immediate need to guard against a bad supply situation.

The shortages extend beyond the TTL family and affect linear devices such as the LM324 quad op amp.

Where necessary manufacturers are trying to design around the TTL content of products and make use of CMOS devices which are at present not affected by the problems blighting TTL. The inherent difference between these two families, particularly in terms of speed, means that this is often not a practical proposition.

In the view of Watford Electronic's Nazir Jessa the shortage of TTL ICs is likely to continue for at least the next three months and during this time industry and constructors at home must be prepared to put up with higher prices and problems of getting hold of the devices required.

A small increase

The price of *E&CM* has been held at 85p for some time now and the fact that we're saying this has probably alerted you to the fact that we're going to have to ask you to pay a little more for the magazine from next month. The price increase will be only 5p; that's about the rate of inflation for the past year. At 90p *E&CM* is still a lot less than the price of a packet of cigarettes and indeed less than the price of a pint of lager in many London pubs. We've managed to keep the price down to this level in spite of considerable increases in our costs. In particular the cost of the paper on which *E&CM* is printed has risen due to the intervention of the Americans in the world paper market. Still all this has nothing to do with the world of electronics and computing. We just hope that you agree that *E&CM* will still offer value for money at 90p.

Value for money

Elsewhere in this issue we review the latest low-cost personal computer from Amstrad. This looks like being the year in which a number of companies that have to date been operating in various sectors of the electronics consumer market place will enter the computer battlefield. The Amstrad looks as it it has been designed by a committee, but far from the negative connotations normally associated with this approach to design, in the case of the Amstrad computer the result looks set to succeed. Amstrad seem to have incorporated all the aspects featured in today's best selling micros and put them together in a package that costs far less than any system that may be seenm as a rival.

Another manufacturer associated with the audio industry is to launch a computer in the very near future but we are sworn not to reveal the company's name at this stage. Make sure you get next month's copy though when we will be able to reveal all.

GARY EVANS

Clots claim to fame

The gaff of the month prize goes to a clear winner, Radolfin Electronics, manufacturer of the Aquarius computer. Their proud boast in an advertisement in the April issue of *Your Computer* was: 'What other computer includes a Z80 microprocessor?'

Answers on a postcard please to Radolfin Electronics, Hyde House, London NW9.

Seconds out for Acorn

At last. Acorn's 6502 BBC second processor is now available. This is the first 6502 number two for the BBC, and it joins the Watford Z80, Cambridge Microsystems Flex running 6809 and Torch Z80 disc pack already on the market (and of course the Torch 68000 Unix pack mentioned elsewhere on this page).

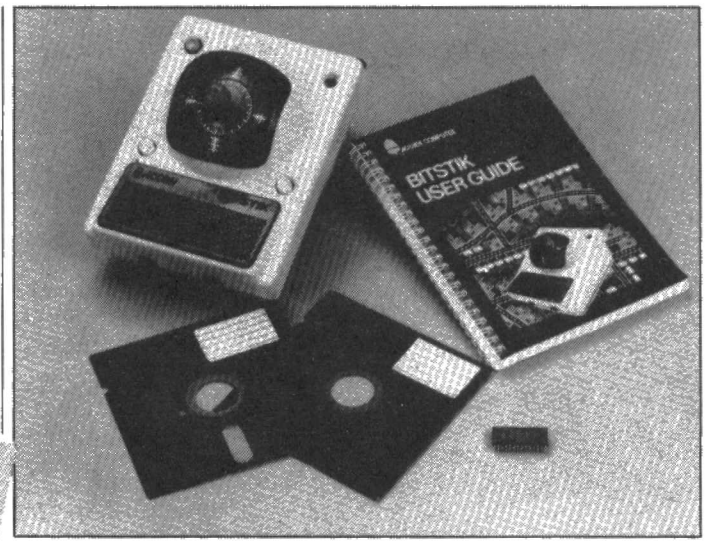
The 6502 2nd processor makes possible many sophisticated applications – particularly those requiring complex graphics or fast handling – that hitherto would have been impracticable on the BBC.

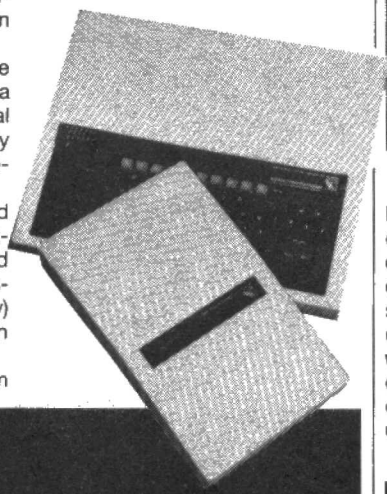
The device uses Acorn's Tube interface: this is a high speed data channel designed to enable external processors to communicate directly with the BBC micro's own processor (also a 6502).

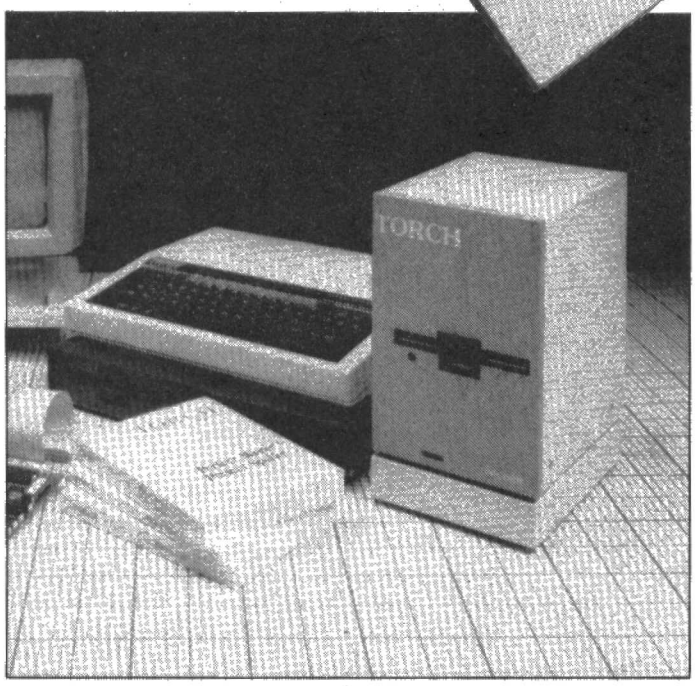
The host processor is dedicated entirely to handling input and output, screen display memory and system filing, while the 2nd processor (with its own 64K memory) simultaneously concentrates on running the application program.

To enable users to get maximum benefit from the 2nd processor Acorn have developed a special 'Hi-BASIC' ROM chip, which frees 44K of user memory for BASIC language applications. Up to 60K is available for assembly language programs. In addition, the full character set can be redefined without using any program memory.

The unit, complete with two ROMs (one to update disk or Econet filing systems) costs £199.







Bitstik is Acorn's first application for the new BBC 6502 2nd processor. Bitstik is a sophisticated CAD system which utilises the extra memory and speed made available by the new processor. It allows lines, style, colour, zoom, manipulation of individual elements and unlimited storage of user defined characters. The system facilitates the drawing of extremely fine and accurate machine drawings on the BBC. The major drawback of the system is that the communications software for dumping screens onto a four-colour flat bed or drum plotter does not yet exist. Bitstik is therefore next to useless for the serious draughtsmen unless he can write his own protocols. The system includes Joystick, ROM-based software, utility disks and manual, and is priced at £375.

Upgrade's new upgrade

Acorn have announced plans to release a Z80 processor for the BBC in the near future, but Upgrade Technology have become the third company to beat them to it.

This device communicates with the BBC via the serial I/O port. It has its own internal floppy disk controller (ie no need for a DFS) and is compatible with both models A and B.

Main advantage of the Upgrade device is that it supports true CP/M and Turbo DOS. This will enormously increase the range of software available to BBC users. The disk controller will handle two drives which can be independently installed by the user via a software utility. The drives can be 3", 5" or 8", double or single sided, 40 or 80 track.

The unit is manufactured by Rade Systems, who formerly built the Watford Electronics BBC Z80 2nd processor. The Upgrade version includes the Rade expansion bus, which allows the user to plug in up to three of Rade's existing add-on boards (including a RAM expansion from 64 to 256K, and serial, parallel or IEEE interfaces).

The upgrade unit will be available towards the end of April and will be priced at £299 plus VAT. *Upgrade Technology, 270a High Road, London NW10 2EU.*

Torch supply BBC UNIX

The extremely popular microcomputer operating system, Unix, is now available on the BBC micro within Torch's Unicorn hard disc pack.

At the top of the range is the 68000 hard disc pack. This unit contains a 68000 16-bit processor, a Z80B for running CP/M programs, 256K bytes of RAM, a 20Mb hard disc, and a 400K floppy disc. Anyone buying it will have to have a very serious and profitable application, for the retail price is similar to that of a good second-hand car: £2,900.

Torch claim that the hard disc pack is the world's least expensive UNIX system. According to the company 'it is expected that the Torch 68000 with UNIX will rapidly establish itself as the leading high-end upgrade for the BBC micro'. This is without doubt true because there is currently no competition, but it will be interesting to see how many amateurs can afford this Ferrari of the home computer world.

Quest for the QL

The QL saga is set to run and run.

Two months after the launch the nearest any journalist has got to a hands-on is at an Open Day given by Psion, writers of QL software.

The estimated time of arrival of QL review machines will be given on April 2nd according to Sinclair spokesman Bill Nichols, and the first general orders will be met 'by the end of the month' (March).

Sinclair have now admitted that they launched a machine which was not ready for sale. At the launch on 12 January both the QDOS operating system software and one of the semi-custom ICs had faults in their design. It is still to be confirmed that these faults have been ironed out.

We cannot promise that the next issue of *E&CM* will carry a full review of this elusive machine, but there is now a reasonable chance that it will appear (Sinclair willing).

Meanwhile back on planet Earth, it has been observed that the Spectrum Microdrive has found its way into the High Street. Sinclair made the announcement in early March that Microdrive would be in the shops by April but untypically, some devices have sneaked into Smiths, Boots and Menzies etc. ahead of schedule.

HI-FI TO HI-RES

An exclusive preview of the new colour computer from Amstrad. Gary Evans reports.

The CPC464 colour computer from Amstrad looks set to shake up the low cost home computer market. The computer is to sell for £199.99 and for that price will provide a complete 'ready to plug in' package. In the past the ready to plug in tag has been applied to machines that require the addition of a cassette recorder or, at very least, the family's TV set, before they can be used. In the case of the Amstrad computer however the £200 purchase price includes a cassette recorder **and** a 12" green phosphor monitor (for an extra £100 Amstrad will supply a colour monitor). In addition the CPC464 provides 64K of RAM, thus the machine can be seen to offer extremely good value for money.

It is this philosophy of offering products engineered down to a price that has led to Amstrad's success in the low end of the Hi-Fi market. The company's products have never been designed to 'set the world of technology alight' but have retailed at prices that have undercut similar products by a considerable margin. In the context of the home computer market the new Oric Atmos, the Dragon and even the Commodore 64 look vulnerable in terms of the features and price of the 464 package.

The machine was designed in the UK yet will, in common with other Amstrad products will be manufactured in the Far East. It should be on sale by the middle of May and is expected to first appear in the traditional outlets for the company's product range, namely the High Street electrical chain stores.

Hardware highlights

The choice of a Z80 processor sets the tone for the design of the computer, a design that adopts tried and tested techniques – no 32 bit processors to be found in the CPC464. The sound facilities of the computer are built around the familiar AY8910 IC, the three sound channels appearing as left, right and centre on the stereo output jack although the internal speaker produces a mixed mono output.

The 64K of RAM supplied as standard on the machine makes over 42K available for BASIC programs thanks to the use of ROM overlay techniques when using BASIC. The 16K BASIC interpreter resides in the top 16K of the 464's memory map and there are facilities in the firmware to call up to 240 additional ROMs by employing a certain amount of address decoding external to the computer.

The styling of the 464 owes something to the hi-fi products with which the company is associated and certainly the inclusion of a built in cassette recorder was a logical step for Amstrad to take. This means that the problems of connecting up a string of cables and the complications of level setting associated with many of the present home micros should not trouble the 464. Recording of data takes place at software selectable rates of either 1K or 2K baud with the read speed being automatically established by the software.

The keyboard is a full feature 'typewriter-style' design complete with a cursor cluster and a standard numeric keypad. Up to 32 of the keys may be redefined for operation as function keys. The user defined functions may consist of up to 32 characters and the redefinition capability includes the auto repeat parameters.

CPC BASIC

The BASIC is described by Amstrad as an industry standard version of the language that is both fast and versatile. The PCW benchmarks show that the 464 lives up to these claims, the average value of the timings shows that the computer is only slightly slower than the BBC micro. The BASIC does not support procedures although to be fair to Amstrad, procedures do not form part of the specification of the BASIC language. As if in response to the lack of procedures though, at the time of launch, Amstrad will make available a PASCAL compiler that owes a lot to the well received version available for the Spectrum.

Screen modes

There are three modes of screen operation: the normal mode offering 40 columns x 25 lines with 4 'ink' text or 320 x 200 pixels individually addressable in 43 colours; the multicolour mode produces a 20 col x 25 line display with 16 'ink' text or 160 x 200 pixels addressable in 16 colours; in High Resolution mode the text display is 80 col x 25 lines with 2 'ink' text or 640 x 200 pixels addressable in 2 colours. The border can be set to any pair of colours (black being treated as colour) and it may either be flashing or steady. Each ink can be set to a pair of colours, ie flashing, or a single steady colour. Text writing can be set to either translucent or opaque, in other words it will either ignore the paper colour and overwrite the graphics or it will completely overwrite the background.

It is possible to select up to eight text windows ('text streams') into which characters are written and also a graphics window into which plotting may be performed.

Ins and outs

The 464 features a joystick port and Centronics compatible printer port as standard. Other hardware expansion will be available via jump blocks or indirection operators to allow software expansion facilities.

Amstrad plan to introduce a number of add-ons for the computer including disc drives and serial interfaces complete with driver software in ROM.

PCW Benchmarks

BM1	1.45
BM2	3.6
BM3	9.7
BM4	9.5
BM5	10.55
BM6	19.65
BM7	30.8
BM8	34.54

Average 14.97.

The CPC464 at a glance

Price	£199.99
Processor	Z80
Keyboard	"Typewriter style" with cursor cluster. Up to 32 keys may be redefined for operation as function keys.
Memory	64K. Over 42K available for BASIC programs.
Storage	Built in cassette recorder.
Graphics	Mode 1 40cols x 25lines. 4 'ink' text Mode 2 20cols x 25 lines 16 'ink' text Mode 3 80cols x 25 lines 2 'ink' text
Language	'Industry standard' BASIC including new commands AFTER and EVERY.
Sound	White noise source. 3 sound channels (each of which can be independently set for tone and amplitude.
Expansion Interfaces	Disc drives, serial interfaces. Joystick port, centronics printer port.

BYTE BACK

Send your letters to The Editor, E&CM, 155 Scriptor Court, Farringdon Road, London EC1R 3AD.

Blowing EPROMS

Sir,

Your devoted attention to BBC micro projects has made me dedicated to your publication. In particular the EPROM blower, probably your most popular project to date, was excellent and mine is working fine incorporated in a Eurocard rack system.

I am writing to enquire whether you have any plans to publish any further articles on the EPROM programmer project and if so could you investigate the additional facility of blowing 2732 EPROMS. I realise the circuit was intended for 8 and 16K devices but the user writing his own ROM routines hardly ever fills 8K, let alone 16K.

So how about a 4K facility for the already well known *E&CM* EPROM programmer?

Yours hopefully,

A. M. Robbie
Edinburgh

We don't have any plans to continue the BBC EPROM articles, but you may have noticed the project in the last issue of E&CM for a 68705 universal EPROM blower. This board can blow 2732 and 2716 devices, with 4K and 2K of memory respectively. (Ed.)

Teletext editor

Sir,

A few errors were made in my Teletext Frame Editor in the April issue of your magazine. The first occurs on lines 180 and 220, both of which should terminate as follows:

```
:PROCend:END
```

Secondly, character 255, the block or inverse space cannot be obtained from the editor as it stands. This is very easy to correct. Change line 1140 as follows:

```
1140  IFc%>1 ANDc%<32
      THEN 1180
```

Then add a new line:

```
1145  IFc%=1 c%=255
```

These alterations are very simple to make. Finally, a bit of confusion may have occurred over the versions of this editor available. There are three versions: the BBC Basic one that was printed in *E&CM*, a BCPL version that I use privately, and a full 6502 machine code version available via Micronet 800. This final version has many enhancements, including full BREAK/CTRL-BREAK protection, screen copying and screen swapping, and of course the extra speed inherent in machine code programs. It is also quite cheap!

Adam Denning

All psyched up

Sir,

I have just read the latest edition of your magazine. I have been a regular reader for some time and I appreciate the technical articles, which I find interesting and informative.

Great was my disappointment then, when I read the article on the Centronics Printer Buffer. It was something and nothing. If you had not advertised it in "Next Month" I would not have been so het up. But if you look at it, there is nothing practical. I am left with no idea what IC's are needed, there is no circuit diagram, there is no picture, there is no information about power requirements. I cannot get any components together or do anything! If you look at the Spectrum real time clock article, at least anyone interested can see what may be involved.

I know that it is your policy to "break" articles. In this case, it would have been better to have described the marvellous way of using the R-register in part two, only alluding to it in the first. The circuit diagram and maybe the PC board could have been in part one. leaving the software for the second part. At least I, and maybe many others would have been able to have got the hardware under way ready for part two next month.

I want to say that I have never written to any Editor complaining about what I have read, but in this case I had got myself all "psyched up" and then there was nothing. I just do not know how I can manage to hang on for another month. Perhaps I had better give up and tend the soil to keep hold of my sanity or perhaps I will go and knock my head against a brick wall!

M. D. J. Foreman
Bristol

Sinclair service

Sir,

Re your article on page 8, of the March Issue.

I ordered my QL on 6-2-84 and shall tell you when it arrives. My only other association with Sinclair was when I ordered the kit version of the ZX81 in August of 1981.

It was delivered promptly and the back up service was as good as I needed. I fumbled the keyboard connection and cracked the conductors on the tails. Sinclair were very sympathetic and sent me a f.o.c. replacement immediately.

A year ago one of the IN4148's controlling the keyboard failed and cost me 2p and ten minutes to replace; but that was the only component failure. Naturally, I had problems with wobble on the 16K extension and obtaining a compatible cassette recorder, but considering the price, I wasn't paying for perfection.

On the whole I shall be satisfied with the QL if it lives up to its expectations as well as the ZX81 did.

C. A. Bearfield

ZX sound board

Sir,

I was unaware of any problems with the Spectrum sound board (August '83) until receiving the letters you forwarded to me. I have been in touch with both correspondents and between us we have eliminated the errors.

There were four errors in the circuit as published which have been corrected (see diagram).

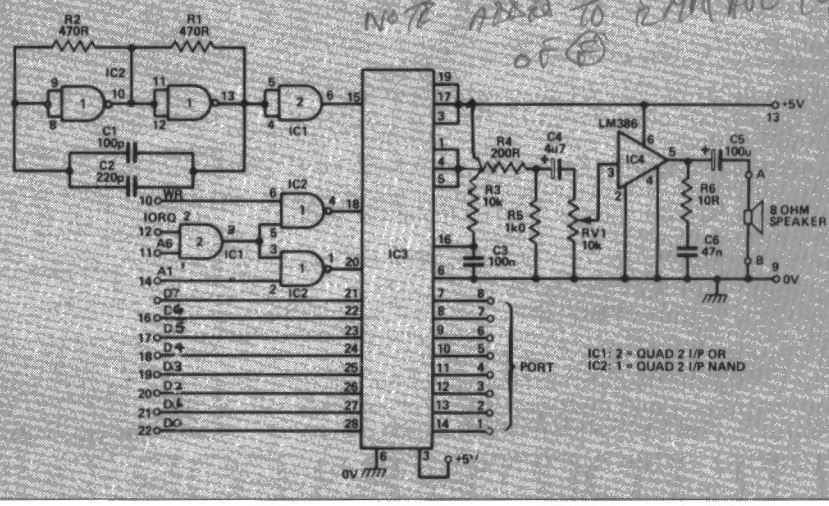
1) IC1 and 2 were marked in reverse in circuit board and overlay.

2) IC1 is 74LS02 and goes in SKT nearest sound chip. IC2 is 74LS32 and goes in SKT nearest edge. D0-7 connections to AY-3-8912 were reversed.

3) PCB layout of IC2/1 is incorrect as pins 1 and 2 are I/Ps and pin 3 is the O/P. This can be rectified by cutting the track to pin 3 and the track to pin 1 thus isolating both. Join the A6 to pin 1 and join pin 3 on IC2 to track going to pins 3 and 5 on IC1.

4) Disconnect R3 from pins 1, 4 and 5 of sound chip and connect it to +5V.

J. McAinsh



MEMEX

Brian Alderwick and Peter Simpson have designed a memory expansion board which adds 20K of RAM to a BBC model B micro – allowing hi-res graphics to be combined with large programs.

Since its public debut two years ago the BBC computer has been hailed as an extraordinary success. Two of its greatest features are the ability to retain a large program and the provision of a very high resolution graphics capability. Unfortunately these two features are mutually exclusive! You can have either but not both. This problem was forseen by Acorn who fitted the TUBE interface to the BBC micro so that a second processor could be attached, enabling large programs, resident in the second processor, to use high resolution graphics in the screen memory of the BBC micro. A second processor is however, an expensive way of providing extra memory.

This article describes 'MEMEX', a 'plug in' addition to the BBC micro which will allow a 29K program (tape) or a 26K program (disc) to use any graphics mode. In addition a listing of the paged ROM software necessary for control will be given.

Operation of MEMEX is completely transparent to the user and it will function correctly with all paged ROMs which follow the operating protocols laid down by Acorn for writing to the screen. On power up, MEMEX will be enabled, allowing large programs and high resolution graphics to be used simultaneously. A single command will disable MEMEX returning the BBC micro to its original state to allow the use of programs which access the screen directly. MEMEX will then remain disabled even through a control/BREAK, until a second, enabling, command is given.

MEMEX is compatible with, amongst others, BASIC 1, BASIC 2, BCPL, HCCS FORTH and VIEW. No increase in available memory is possible with WORDWISE since it uses a fixed workspace for its text. MEMEX requires a BBC B and is compatible with OS 1.2 but has not been tested with OS 1.0 and will not work with OS 0.1.

General description

MEMEX plugs into the 6502 microprocessor socket in the BBC micro and is accompanied by a paged ROM which contains the code necessary to operate it. No soldering to the BBC micro is required. In operation the extra memory carried by MEMEX is decoded to lie between addresses &3000 and &7FFF. This is the area normally used by the highest resolution screen modes whilst lower modes use subsets of this area. When MEMEX is enabled and languages such as BASIC or VIEW access the screen memory area then they will access the memory carried by MEMEX. However, when a character is sent to the VDU driver for printing to the screen, the MEMEX memory is switched out before printing the character and subsequently switched back in. Thus the character will go to the screen memory resident in the BBC micro rather than to MEMEX. The process of switching the MEMEX memory in and out is controlled by extra code attached to oswrch, the write character operating system routine, via its vector at &20E.

The selection is dictated by the setting of a two-state latch. This latch can be set or cleared by writing to the addresses &C000 and &D000 respectively. These addresses are in the operating system ROM. This method was chosen to simplify address decoding. Writing to a ROM is a meaningless act, from a programming point of view, and the authors are not aware of any program for the BBC micro which does this. The only time when problems could arise is if a file is *LOAD'ed onto the OS ROM to verify the correct SAVEing of the program. This procedure should be avoided when MEMEX is present.

Several new commands are provided in the controlling software. Specifically there are commands to enable and disable MEMEX (*MEMIN and *MEMOUT), to load or save the screen directly from the current filing system (*SCREENLOAD and * SCREENSAVE) and to enable access to the screen memory without disabling the MEMEX system (*FX100). In addition the command *MEMTEST is provided which will verify that the MEMEX memory is functioning correctly.

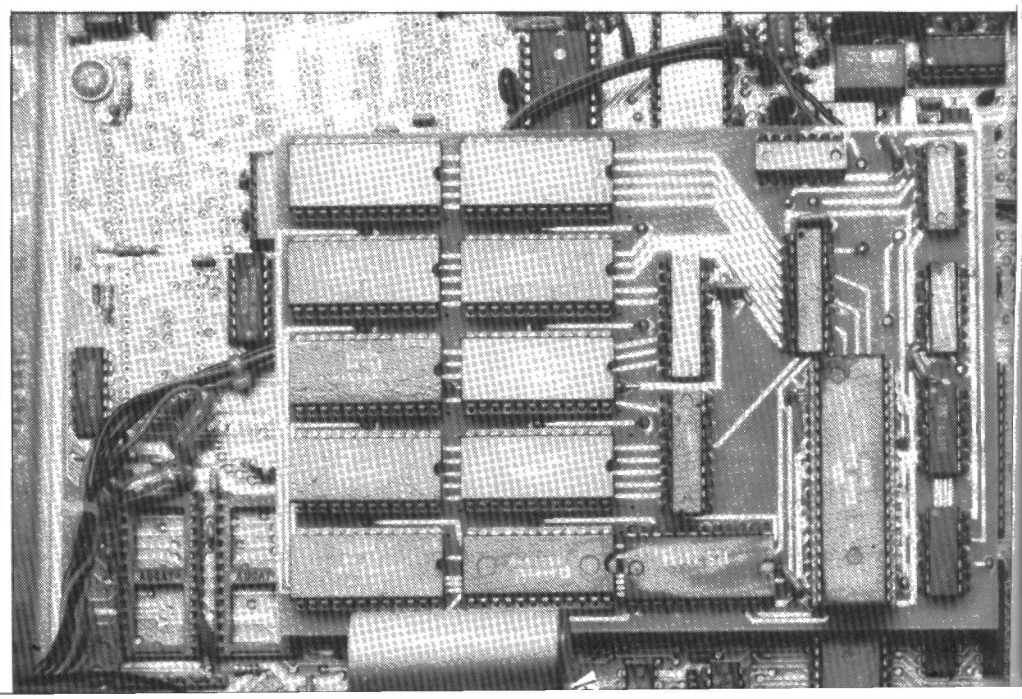
The value of the BASIC variable PAGE is unchanged when MEMEX is present in the computer but not activated and PAGE may be set below the value assigned after a BREAK. However, when MEMEX is activated, PAGE will be &100 higher than normal after a BREAK and it should never be set lower than this value since the MEMEX operational code will be overwritten and the computer will crash. Despite the higher value of PAGE, there will still be an increase in program space of &300 bytes in MODE 7.

Hardware details

The hardware, whose circuit diagram is shown in **Figure 1,** can be divided into three sections: the 20K block of random access memory, the latch to control which bank of memory is accessed and the RAM address decoding section together with the read/write control, phi 1 control and the data bus buffer.

The RAM memory is of straightforward design consisting of ten 2K devices, IC 10 to IC 19, each of which is enabled by one output of a four to sixteen line decoder IC 9. The decoder is enabled by the bar20 output from the address decoder section and by clock phi 1 being low. This ensures that the RAM chips are only enabled when valid data is present on the data bus. The data lines of the RAMs are connected in parallel and go to a bidirectional buffer IC 6 to minimise loading on the 6502.

The latch section comprises two parts, IC5, an edge-triggered D type flip-flop with additional clear and set inputs and address



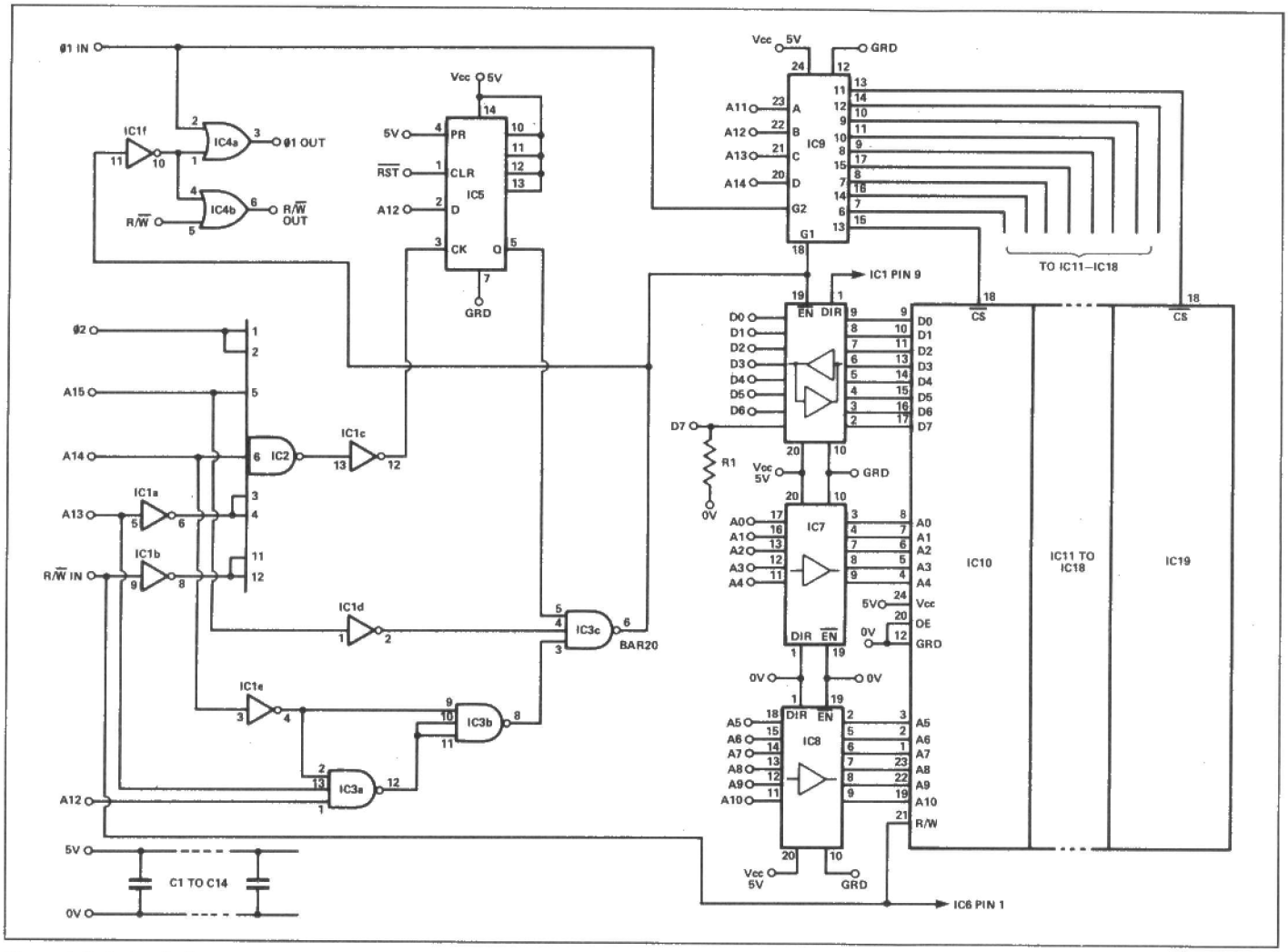
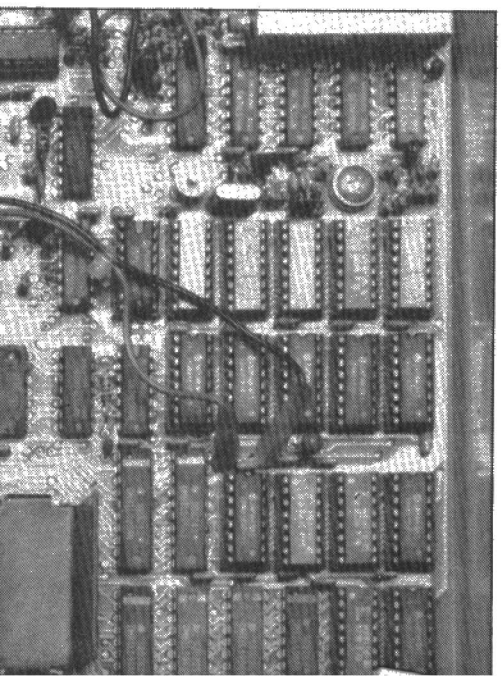


Figure 1. Full circuit diagram.

decoder IC2. This is an 8-input NAND gate, of which five inputs are used. The output will go low when three conditions are fulfilled. First, the address must be in the range &C000 to &DFFF, secondly, the 6502 must be WRITING data to that address and thirdly, phi 2 must be high, which only occurs when the address is valid. When these conditions are fulfilled the output of IC2 goes low and this is inverted by IC1c. This low to high transition is applied to the clock input of the flip-flop and causes the state of the D pin to be copied to the Q output. The state of the D pin is determined by address line A12, thus an address of &C000 sets the Q output low and the address &D000 sets the Q output high. At any time the reset line is able to set the output of the latch to the low state via the clear input of the flip-flop.



'Operation of Memex is completely transparent to the user'.

The address decoding, phi 2 and read/write section comprises IC3 and IC4. to enable the MEMEX RAM, via IC6 and IC9, the bar20 signal must be low and this requires two conditions to be fulfilled. First, the address must be in the range &3000 to &7FFF (IC3, IC1d and IC1e) and secondly, the flip-flop Q output must be high. The BBC board is disabled by forcing both the phi 1 and the read/write signals to the high state. This is achieved, via two, 2-input OR gates IC4a and IC4b, the inverted bar20 signal being applied to one input on each of the OR gates. When the MEMEX RAM is activated, the bar20 signal is low, forcing the OR gate outputs high regardless of the states of the phi 1 and read/write lines. When the bar20 signal is high, then a low input is applied to the OR gates allowing the outputs to reflect the states of the phi 1 and read/write lines.

When the BBC board is disabled, it is still receiving addresses from the 6502 which fall in the RAM area and to which it responds. By forcing the phi 1 clock high the buffer between the BBC RAM and the 6502 is disabled, thus, although the BBC RAM is responding correctly, the buffer does not pass on the information to the 6502. This leaves the way clear for the MEMEX RAM to control the data bus via its buffer which is enabled when the BBC board is disabled. The read/write line to the BBC board is forced high as well as phi 1 so that corrupt data is not written into the BBC RAM when data is written into the MEMEX RAM. On the BBC board the phi 1 signal from the 6502 is inverted and used as a 2MHz signal to clock data into numerous chips. If this signal is significantly delayed then corrupt data could be clocked into these chips. This design introduces only one delay into the phi 1 signal, that due to the OR gate. The prototypes have functioned correctly with a 74LS32 chip but, inevitably, there will be a spread in characteristics between different chips and BBC computers which may cause

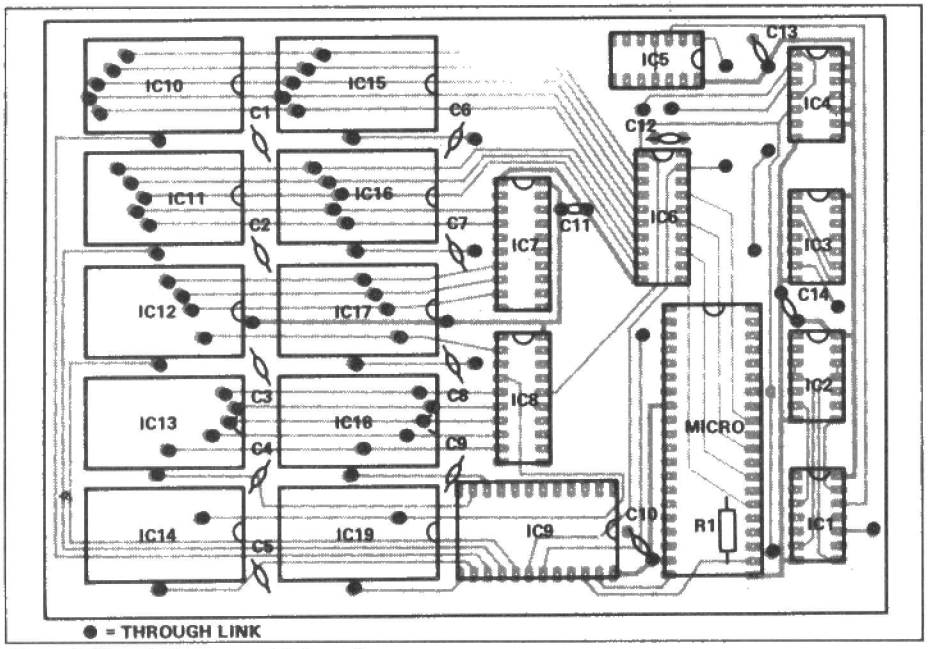


Figure 2. Pinout diagram and foil overlay.

difficulty for some people. Hence IC4 is specified as the faster 74S32 or 74ALS32. The project is constructed on a 108mm x 152mm double-sided printed circuit board. For constructors prepared to make their own boards the topside foil is shown in **Figure 2.** The pads in the memory area are very small and it is recommended that a 0.7mm drill is used for holes to house the sockets for ICs 10 to 19. A printed circuit board is available through the *E&CM* service and details will be found in their advertisement. All the TTL chips are

'Several new commands are included in the software, full details of which will be given next month'.

followed. First, solder the resistor R1 and then all of the thro and decoupling capacitors as shown in **Figure 4,** the component overlay.

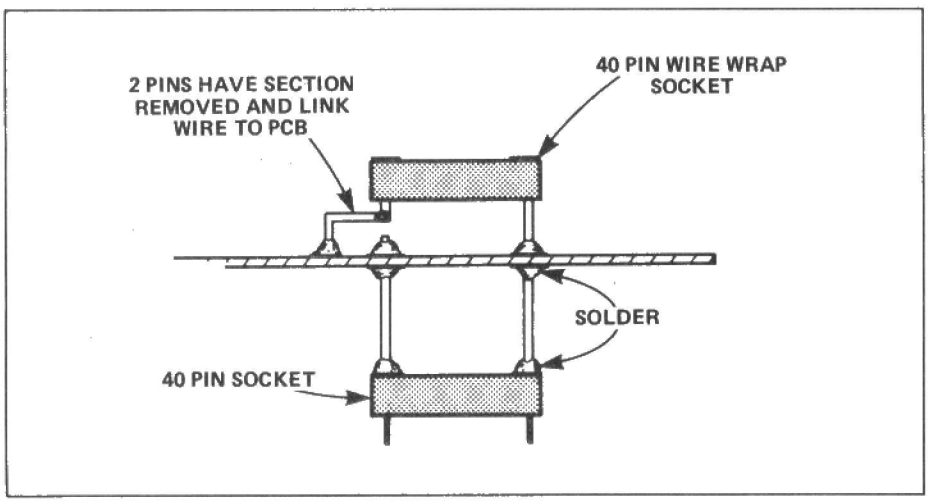


Figure 3. Soldering the capacitors.

soldered on both the top and bottom of the board as are all other positions which have pads on both sides of the board eg capacitors and wire links. The wire links should be trimmed close to the board, especially in the memory area, as the sockets will have to be mounted directly on top of them. Some of the pads, especially around the memory chips, are very close together and care will be needed. A fine point soldering iron is recommended.

To enable the board to be checked at various stages of construction the following assembly instructions should be

Next install the microprocessor socketry. This elevates the board high enough above the components on the BBC main board to avoid fouling. The 40-pin wire wrap socket is fed through the top of the board until the underside of the socket is 5mm above it. Make sure the socket is parallel to the board and solder all 40 connection on both the top and the underside. The pins projecting below the board should then be soldered on the top of a 40-way turned pin socket. This bottom socket can be an IC socket of a solder on DIL header, but it is important that it is a turned pin type as this will ultimately plug into the processor socket on the main board. Experience shows that this kind of socket, used as a plug, does no harm to the processor socket, but that plugging in a wire-wrap socket certainly does.

Returning to the top side of the board, locate pins 3 and 34 of the 40 pin socket. A 2mm section should now be snipped out of these two pins, but be careful to leave enough pin projecting below the underside of the socket so that an L-shaped link can be soldered to the adjacent pads.

Install IC4 and IC5 and make a temporary link from IC4 pin 4 to IC4 pin 7. Check for short circuits! Before MEMEX can be tried in the main BBC board the two power connectors (located about centre) will have to be bent over to give enough clearance. Carefully bend the tag which is attached to the cable through approximately 90 degrees. Remove the 6502 processor from the socket on the BBC main board and plug it into MEMEX and in turn plug MEMEX into the main board. On power up the micro should behave perfectly normally. If so then continue, if not then search for open or short circuits in the microprocessor sockets and around IC4.

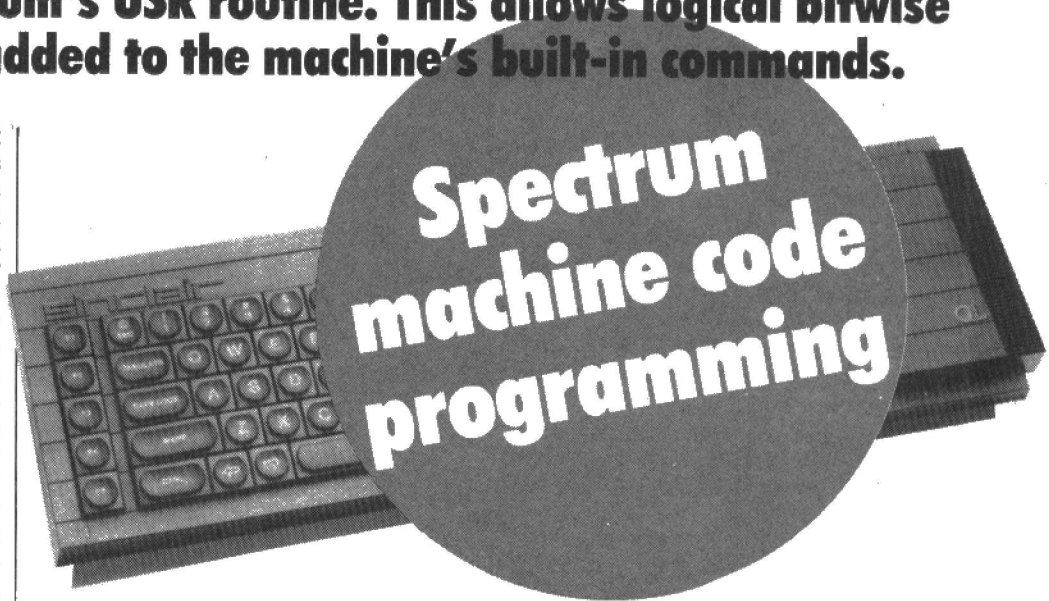
Remove the temporary link from IC4, install IC1 and IC2, insert a temporary link from IC1 pin 11 to IC1 pin 14 and check for short circuits etc. On power up behaviour should again be normal. Measure the voltage at IC5 pin 5. It should be less than 1.0V. If so then enter the BASIC command ?&D000=0 and measure the voltage again. It should be greater than 3.0V. The BASIC command ?&C000=0 should return the voltage to its original level. If the voltage does not respond properly then checks should be made in the vicinity of IC1, IC2 and IC5.

Remove the temporary link from IC1 and install IC3, IC7, IC6, IC8 and IC9 one at a time in that order testing the computer for functionality between each addition. Finally, solder in the socket for IC12 and insert the memory chip. Refit the MEMEX board to the micro, plug the MEMEX EPROM into a paged ROM socket, hold your breath, and switch on. The MEMEX logo should appear after the BBC Computer 32K message. If it does then install the rest of the RAM sockets and IC's and try *MEMTEST. This will confirm the operation of the MEMEX RAM and will show up any hardware faults in the address lines by printing the memory block start address and the address line number eg A5 followed by a question mark. If such a message should occur then careful checks should be made for continuity and for short circuits. If the MEMEX logo does not appear then IC6, IC9 and IC12 should be carefully checked for short or open circuits in particular the data and chip select lines.

Next month – Full details of the sideways ROM software to support the MEMEX hardware.

The logical Spectrum

Mike James describes a technique that allows parameters to be passed into a Spectrum's USR routine. This allows logical bitwise operations to be added to the machine's built-in commands.



The ZX BASIC on the Spectrum doesn't implement the standard logical operators AND, OR and NOT in the same way as other versions of BASIC. This isn't a serious problem unless you want to work directly with the Spectrum's hardware or use 'bit manipulation' to separate information stored within a single memory location. In principle the solution to this problem is simple, all you have to do is write a few short machine code routines that perform the 'bitwise' versions of the logical operations provided by other dialects of BASIC. In practice the problem is in using these routines from BASIC. For although the method of calling a machine code routine from BASIC as a USR function makes provision for passing a result back it makes no provision for passing parameters into the routine. However using a little knowledge of how ZX BASIC works it is possible to pass any number of parameters to a machine code routine and hence add new functions that are almost treated in the same way as the standard built-in functions.

'The method of calling a machine code routine makes no prism for passing parameters'.

The first part of this article explains in more detail why bit-wise versions of the logic functions are useful. Then a method of passing parameters to USR functions is decribed in general terms and finaly three new functions are created equivalent to AND, OR and NOT. The three new functions are useful even if you only use ZX BASIC, but if you write Z80 machine code the method of passing parameters opens up new possibilities for combining BASIC and machine code.

Bit manipulation

One of the common features of programs that use a machine's hardware directly is the need to resort to 'bit manipulation'. The reason for this is that the state of a particular bit or group of bits often reflects or controls the condition of some hardware. Another reason for wanting to examine and change bits or groups of bits is the use of different parts of a byte to hold different pieces of information. For example, an attribute byte uses b7 for flashing on/off, b6 for bright on/off, and b5 to b3 and b2 to b0 for paper and ink colours respectively.

In other versions of BASIC bit manipulation is performed using the logical operators AND, OR and NOT but in ZX BASIC these operators behave differently. In normal use in ZX BASIC these operators work with the values 0 and 1 representing false and true respectively. For example, the result of x AND y is 1 if both x and y are 1 and 0 if either of them are 0. This corresponds to the usual English interpretation of AND that 'x and y' is true only if both x is true and y is true. However, ZX BASIC interprets any non-zero value as true and this gives rise to the following results when x and y are other than 0 or 1:

x AND y = x if y is non-zero
= 0 if y is 0
x OR y = 1 if y is non-zero
= x if y is 0
NOT x = 0 if x is non-zero
= 1 if x is zero

These results are useful for writing conditional expressions such as described in Chapter 13 of the Spectrum Manual and in "In praise of ZX BASIC" *E&CM* October 1983 but they are not suitable for bit manipulation.

The sort of implementation of AND, OR and NOT that other versions of BASIC have are 'bitwise' operations that are much more useful in bit manipulation. For example, the result of a bitwise AND operation is arrived at by ANDing the corresponding bits in each of its operands. That is b0 of the result is arrived at by ANDing b0 of the first operand with b0 of the second and so on. Thus the result of a bitwise AND of 7 and 12 is –

7 = 00000111
12 = 00001100
7 AND 12 = 00000100

or 4 in decimal but the Spectrum's AND operation gives the result 7.

The importance of bitwise operations for bit manipulation is that you can set any bit or groups of bits to zero by ANDing them with a 'mask' value and you can set any bit or group of bits to one by ORing them with a mask value. To be precise –

to set any bits to zero construct a mask value consisting of ones in every bit position apart from the bit positions that you want to set to zero. This mask should then be bitwise ANDed with the value that contains the bits that are to be set to zero.

to set any bits to one construct a mask value consisting of zeros in every bit position apart from the bit positions that you want to set to one. This mask should then be bitwise ORed with the value that contains the bits that are to be set to one.

For example, to set bits b7 to b4 to zero in any value it would have to be bitwise ANDed with –

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	1	1	1	1

ie 15 in decimal. To set bits b7 and b6 to one the value would have to be bitwise ORed with –

b7	b6	b5	b4	b3	b2	b1	b0
1	1	0	0	0	0	0	0

ie 192 in decimal.

Of course the trouble with these methods is that ZX BASIC doesn't have bitwise AND, OR and NOT operators.

Passing parameters

The advantages of machine code routines implemented via a USR function are indisputable. However to implement the bitwise logical functions we need to pass parameters to the machine code routine

'there is a way of writing m/c routines to accept standard BASIC parameters'.

and then have it pass back the result. In fact USR functions always return a result – the 16 bit number stored in the BC register pair. For example, the program

```
LD BC,42
RET
```

will return the value 42 if called as a USR function. The problem that has to be solved is the passing of parameters to the routines. The most obvious and most widely used method is to employ fixed memory locations as 'post boxes'. A post box is used to pass data to machine code user routines by POKEing it into the locations before calling the routine with USR.

This works but it isn't very flexble and doesn't fit in with the way other functions work.

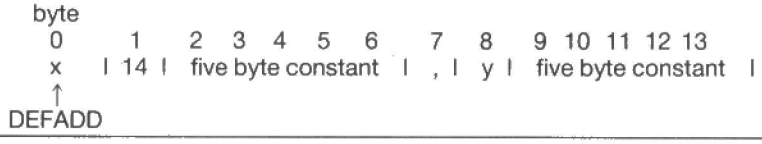
There is a way of writing machine code routines so that they accept standard ZX BASIC parameters. The method relies on building the USR call into a user-defined function with the required number of parameters. For example, if you want a machine code routine that will add two 16-bit positive numbers together you could define a function

```
DEF FNa(x,y)=USR 23296
```

assuming that the machine code is stored in the ZX printer buffer. The only problem that remains is how the USR function is to gain access to the values of the parameters 'x' and 'y'. The solution lies in the system variable DEFADD (23563) which contains the address of the first parameter of a user-defined function while the function is being evaluated. Thus, in the program

```
10 DEF FNa(x,y)=USR 23296
20 PRINT FNa(2,3)
```

DEFADD will hold the address of the 'x' in line 10 when the function at line 20 is executed. This means that the USR routine can use DEFADD to find the memory location that holds the 'x' in line 10. You may be wondering why knowing the location of the name of the parameter used in a function is of any use in finding its value. The answer is that when a user-defined function is being evaluated by ZX BASIC each of the actual parameters used are themselves evaluated and then stored in five bytes following each parameter name in the function definition. (That is while the program is running the parameter values are stored not within the data area of memory but within the program itself!). This means that each of the parameters is evaluated in line 20 giving the result 2 for x and 3 for y. (Of course in general the evaluation can be much more complicated, involving full arithmetic expressions and other functions). Then the result 2 is stored in the five bytes following the letter x in line 10 and the result 3 is stored in the five bytes following the letter y in line 10. Each of these five bytes is preceded by a byte containing 14, the control code that indicates that a number follows. This stops the parameter values appearing in program listings. Thus at the time that the machine code USR routine is called the format of the last part of line 10 in the program is shown in **Figure 1.**



By using the value in DEFADD the USR routine can easily pick up the values of the parameters.

Although it is possible to write routines that process full five-byte floating point numbers it is much easier if parameter values are restricted to 16-bit integers. A 16-bit integer value is stored in a special format using the second, third and fourth bytes. (This format is described fully in the ZX BASIC manual). In fact if only positive integers are used the 16-bit value can be found in the third and fourth byte of the five bytes.

'Using the bitwise logical functions makes the isolation of parts of a byte very easy'.

The routine to add two 16-bit positive numbers is now easy to write, see **Program 1.** (Notice that it has been assembled to load and run in the printer buffer).

address	assembly language	code
23296	LD IX,(23563)	221,42,11,92
23300	LD A,(IX+4)	221,126,4
23303	ADD A,(IX+12)	221,134,12
23306	LD C,A	79
23307	LD A,(IX+5)	221,126,5
23310	ADC A,(IX+13)	221,142,13
23313	LD B,A	71
23314	RET	201

comment
load IX with the
address of 1st parameter
load A with 1st byte
of 1st parameter
add 1st byte of 2nd
parameter to A
store result in C
load A with the 2nd byte
of the 1st parameter
add 2nd byte of 2nd
parameter
store result
return to BASIC

The ZX BASIC listing shown as **Program 2** loads the routine into the printer buffer and gives an example of its use –

```
 10  DATA 221,42,11,92,221,126,4,221,
134,12,79,221,126,5,221,142,13,71,201
 20  FOR A=23296 TO 23314
 30  READ D
 40  POKE A,D
 50  NEXT A
 60  DEF FNa(x,y)=USR 23296
 70  INPUT A,B
 80  PRINT FNa(a,b)
 90  GOTO 70
```

By entering integer values in response to line 70 you will find that their sum is PRINTed by line 80. You might like to experiment with using FNa in more complicated expressions. For example, change line 80 to

```
 80  PRINT FNa(A,FNa(A,A))
```

to add A to A+A. The point is that this method of passing parameters to a machine code routine results in a function that can be mixed with other functions and used in exactly the same way that they can. Of course adding two 16-bit numbers together is not a very useful operation for a machine code function but the same method can be used to add the logic functions that we require.

Bitwise functions

Now that the method of passing parameters has been described there is almost nothing else to explain! The only extra information needed is that the Z80 has an AND operation that will perform a bitwise AND with the current contents of the A register and a memory location, a similar OR operation and the CPL (ComPLement) will perform bitwise NOT on the contents of the A register.

The following assembly language routine shown in **Program 3** will perform the bitwise AND between two 16-bit integers.

If this AND routine is compared with the 16-bit addition routine given above you will see that the only difference is that the ADD instructions have been changed to AND. In the same way a bitwise OR routine can be produced by changing the two AND instructions to:

```
      OR (IX+12)   221,182,12
and
      OR (IX+13)   221,182,13
```

To complete the set a single parameter 16-bit NOT routine is shown in **Program 4.**

address	assembly language	code	comment
23296	LD IX,(23563)	221,42,11,92	get parameter address
23300	LD A,(IX+4)	221,126,4	1st byte 1st parameter
23303	AND (IX+12)	221,166,12	AND with 1st byte of 2nd parameter
23306	LD C,A	79	store result
23307	LD A,(IX+5)	221,126,5	2nd byte 1st parameter
23310	AND (IX+13)	221,166,13	AND with 2nd byte of 2nd parameter
23313	LD B,A	71	store result
23314	RET	201	return to BASIC

address	assembly language	code	comment
23296	LD IX,(23563)	221,42,11,92	get parameter address
23300	LD A,(IX+4)	221,126,4	load 1st byte
23303	CPL	47	complement (NOT) A
23304	LD C,A	79	store result
23305	LD A,(IX,5)	221,126,5	load 2nd byte
23308	CPL	47	complement (NOT) A
23309	LD B,A	71	store result
23310	RET	201	return to BASIC

Although these three routines have been described as if they were each intended to be loaded at the start of the printer buffer they are in fact position independent and can be loaded anywhere in memory. The BASIC program shown in **Program 5** loads the machine code for all three routines into the printer buffer and defines the three functions

```
          FNa(x,y)   which performs the bitwise AND of x and y
          FNo(x,y)   which performs the bitwise OR of x and y
and       FNn(x)     which performs the bitwise NOT of x
 10  DATA 221, 42,11,92,221,126,4,221,166,
     12,79,221,126,5,221,166,13,71,201
 20  DATA 221,12,11,72,221,126,4,221,102,
     12,79,221,126,5,221,182,13,71,201
 30  DATA 221,42,11,92,221,126,4,47,79,
     221,126,5,47,71,201
 40  FOR A=23296 TO 23348
 50  READ D
 60  POKE A,D
 70  NEXT A
100  DEF FNa(X,y)=USR 23296
110  DEF FNo(X,Y)=USR 23315
120  DEF FNn(X,Y)=USR 23334
130  INPUT A,B
140  PRINT FNa(A,B),FNo(A,B),FNn(A)
150  GOTO 130
```

An example

As an example of how the AND, OR and NOT functions can be used to simplify things consider the problem of separating out the information supplied by the ATTR function. Normally this has to be solved using bit manipulation techniques based on multiplying and dividing by powers of two. Multiplying by two is equivalent to shifting the pattern of bits that represents a value one place to the left and adding a zero to the right. This is equivalent to what happens to the pattern of digits when multiplied by 10. Similarly dividing by 2 and taking the INTeger part is equivalent to shifting the bit pattern to the right and losing the old value of B0. Using these shift operations it is possible to isolate groups of bits within a byte and it is even possible to set individual bits to 0 and 1 but this is usually very involved. Using the bitwise logical functions makes the isolation of parts of a byte very easy. For example, to isolate the ink colour (b2,b1,b0) from ATTR is now simple:

```
ink=FNa(BIN 111,ATTR(line,col))
```

To isolate the paper colour (b5,b4,b3) is just as simple:

```
paper=INT(FNa(BIN 111000,ATR(line,col))/8)
```

Finally, bright and flash are given by

```
bright=INT(FNa(BIN 1000000,ATTR(line,col))/64)
```

and

```
flash=INT(FNa(BIN 10000000,ATTR(line,col))/128)
```

Conclusion

It is difficult to know which is the most useful, the three bitwise logical functions or the method of passing parameters to USR routines! Since developing the method of passing parameters I have found all sorts of uses for it and have even gone back and modified existing machine code routines to take advantage of it. On the other hand I have also gone back to BASIC programs that manipulate system variables etc. and simplified them using the new logic functions. I suppose it all depends on exactly where your interests lie! However it does show that it sometimes pays off to dig around inside ZX BASIC! ■

Centronics printer buffer

Part 2 of Robert Harvey's printer buffer project has all the necessary software and circuit information.

This buffer is intended for use with parallel printers only (ie: printers interfaced by something like the Centronics standard interface). Therefore an 8-bit parallel input port with strobe, acknowledge, and busy protocol lines and an 8-bit parallel output port with protocol lines are needed. It would, of course, be possible to make this with standard logic: this would require a fair number of chips and would be quite expensive. The other alternative is to use a parallel interface adaptor chip. The Z80 PIO was considered but was found to be lacking enough i/o lines to implement both input and output functions. The chip chosen was the Intel 8255, this has 24 bits of I/O ports and contains the necessary logic to implement the handshake protocol automatically. **Figure 5** shows the timing of

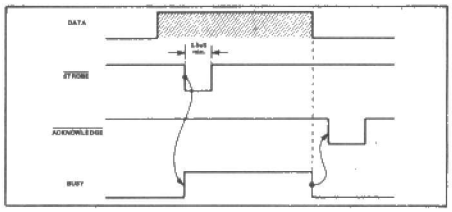


Figure 5. The handshake protocol.

the handshake protocol. You should note that the strobe signal on the parallel input port can be very short indeed, too short for the microprocessor to detect it simply by polling the I/O ports. This is where the 8255 helps, it latches the strobe pulse internally and indicates that fact by raising an interrupt bit. When the CPU eventually gets round to polling the 8255 and reading the input data, the 8255 then generates an Acknowledge signal and resets the interrupt bit. The only thing not fully supported by the 8255 is a pulsed strobe output for the parallel output port; this is overcome by using a spare address line (A13) as a strobe output (after inversion). Thus when an I/O instruction is issued with this address bit high a 2us pulse is produced on the address output. This address bit never goes high at any other time. By now you will have realised that anything goes when using micros as components rather than as computers!

The driving software

The software that drives the printer buffer is held in a 2716 2K EPROM. Although the code itself is just over 512 bytes long, the 2716 is a convenient chip to use in that it requires only a single 5V power supply and it has two chip select inputs. It is worth noticing that although the circuit contains the DRAM required for storing the buffered data, it has no scratchpad memory for the CPU to use. This means that the Z80 must hold all its data in its internal register set. This is not too much of a problem as the Z80 has 208 bits of usable internal storage although even the stack pointer (SP) is used for arithmetic! Another consequence of this is that no subroutine calls can be performed, but as the buffer software is fairly linear code this doesn't matter too much. In fact the software uses a simple form of call where the IV register holds the return address.

The software itself consists of several parts: Initialisation of the system ready for action; Determination of how much DRAM memory is present; polling the input and output and buffering the data.

Initialisation

This is very simple and consists of setting up the 8255 for Mode 1 I/O operation and initialising a few registers with pointers to the DRAM area.

RAM size determination

As I said earlier, the price of DRAMs makes it desirable for the user to be able to start off with a small amount of DRAM, say 8K bytes (1 chip), and add more as he can afford it, up to the limit of 64K bytes (8 chips). This is not quite as simple as it sounds as each chip is only one bit wide so each byte input into the buffer must be broken up into bits and these fed serially to the DRAM. The reverse of this must be performed when reading the DRAM. It makes the software simpler if the number of bits present is an integral factor of eight, thus the following memory sizes are allowed:

1 chip = 8K bytes
2 chips = 16K bytes
4 chips = 32K bytes
8 chips = 64K bytes

If the number of chips used is less than eight, then they must be put into the most significant bit positions in the DRAM section.

For example, if we want 16K bytes of DRAM then we must insert two DRAMs into the most significant bits of the DRAM section and, although the software will read a byte from each of the 65536 locations in the DRAM section, only the top two bits of each byte will contain useful data. The software will thus be required to read four bytes to assemble one byte of data, but to the user it will appear that he has 16384 bytes and not 65536 x 2 bits of data!

The software determines the memory size by writing zeros to the first byte and reading it back; then tries writing ones and reading them back and by looking at the results it is possible to determine how many bits per byte of DRAM are present. If the software sees no bits present then the Z80 halts, this is useful in debugging the hardware as the Z80 HALT output will be low in the halt condition.

The buffer circuit

Looking at **Figure 6** you may notice an absence of address decoding logic, this is because individual address lines are used to select different devices. The software for the system is held in a 2K byte EPROM (2716) and this is enabled when address line 15 is low and a memory access is performed. This apparent 'waste' of the bottom 32K addressing range of the Z80 is of no consequence because the DRAM is not addressed as it normally would be. The DRAM is selected by address line 14 being high but because the address it receives is taken from the R and L registers (due to the driving software) all 64K of the DRAM can be addressed.

The 8255 is selected by any I/O operation (ie: IORQ going low) and address bits 0 and 1 are used to select one of the four internal registers.

The circuit itself contains only a few other things: The circuit required to generate the RAS and CAS signals for the DRAM; A crystal oscillator to provide the clock for the Z80 and a five volt regulator. The circuit can be powered by a 9V DC mains adaptor such as those used by calculators. Also decoupling capacitors should be used, with at least one for every chip on the board.

Polling and buffering

Once all initialisation and memory checks have been performed the processor begins the main loop of the program and starts polling the input and output ports. If a byte is received from the input port and if the buffer is not busy then the byte will be written to the DRAM, otherwise the byte will not be read from the 8255 (thus the sender will see it as being busy) until the

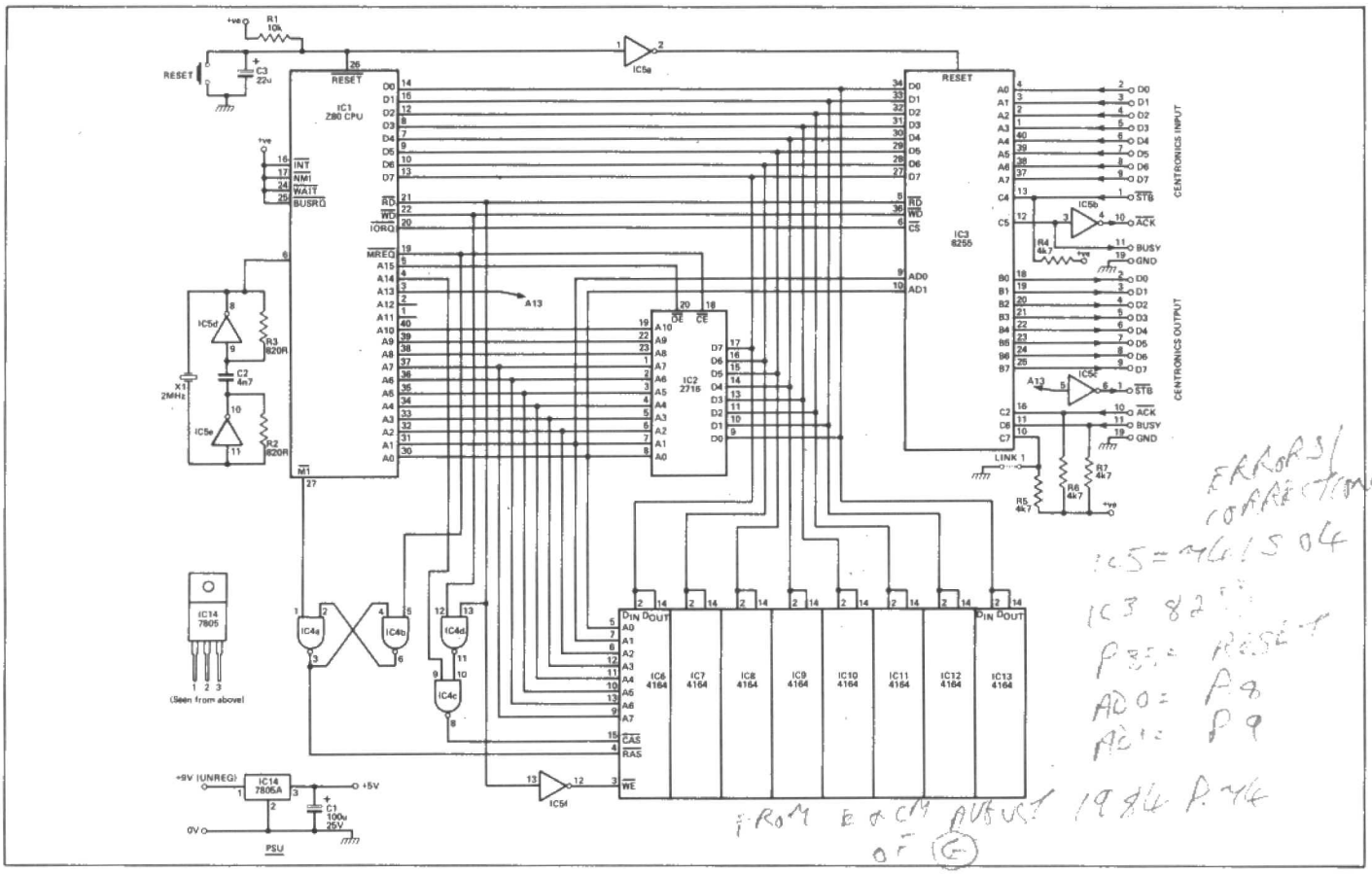


Figure 6. Full circuit diagram of the printer buffer.

buffer has emptied enough memory to become non busy. The software implements a dual threshold busy system. Initially the buffer is not busy but when the buffer becomes completely full it becomes busy and will accept no more data (the computer should then stop sending data). All the time the printer will be taking data from the buffer and once there is more than 256 bytes of spare memory in the buffer the input becomes not busy again. This apparent complication of the input mechanism may not at first appear particularly useful but it does mean that the sender knows that after the buffer becomes not busy it can send at least 256 bytes of data without being interrupted.

"... DRAM may be added in stages ..."

The DRAM is treated as a circular buffer with two pointers to it: one points to the position data may be inserted at (stored in HL') and the other at the position the next data byte may be read from (DE'). These two pointers start off at the same place and are incremented every time they are used. The buffer is deemed full when the input pointer catches up with the output pointer. This is the way the FIFO system is implemented.

Simple self test facility

In order to aid debugging of the constructed unit and diagnose any faults that occur during its use, the buffer has a self

Listing 1. HEX dump of the buffer's software.

```
0000  DD F9 01 03 00 3E BD ED 79 3E 05 ED 79 3E 09 ED
0010  79 21 00 C0 AF ED 4F 77 ED 4F 4E 06 FF ED 4F 70
0020  ED 4F 7E 2F B1 06 08 07 38 02 10 FB 3E 08 90 20
0030  04 76 C3 00 00 06 00 DD 21 6F 02 FE 08 30 1A 06
0040  01 DD 21 69 02 FE 04 30 10 06 02 DD 21 63 02 FE
0050  02 30 06 06 03 DD 21 5D 02 0E 00 21 00 00 11 00
0060  00 D9 01 02 00 ED 78 CB 7F 28 62 21 00 00 39 11
0070  D2 04 B7 ED 52 28 56 21 AC 02 AF 86 2B CB 7C 28
0080  FA 21 00 00 0E 03 B7 28 02 0E 07 DD F9 DD 21 2B
0090  02 DD 7E 00 DD 23 B7 5F FD 21 91 00 C2 D9 01 CB
00A0  41 28 0A CB 81 DD 21 00 00 DD 39 18 E4 CB 49 28
00B0  08 CB 89 DD 21 75 02 18 D8 CB 51 28 08 CB 91 DD
00C0  21 8F 02 18 CC 7D D9 6F D9 7C D9 67 D9 DD 21 D2
00D0  04 D9 B7 ED 52 78 F9 19 D9 21 00 00 B7 ED 72 2B
00E0  47 B7 28 06 CB 3C CB 1D 10 FA EB 21 00 01 AF ED
00F0  52 17 21 00 00 ED 52 17 5F 01 02 00 ED 78 CB 5F
0100  28 33 D9 CB 41 D9 20 0A CB 43 20 0E D9 CB C1 D9
0110  18 23 CB 4B 28 1F D9 CB 81 D9 0E 00 ED 58 D9 7D
0120  D9 6F D9 7C D9 67 FD 21 2D 01 C3 D9 01 7D D9 6F
0130  D9 7C D9 67 D9 D9 B7 ED 52 19 D9 28 2D 01 02 00
0140  ED 78 CB 77 20 24 CB 4F 28 20 D9 7B D9 6F D9 7A
0150  D9 67 FD 21 59 01 C3 7E 01 01 01 00 ED 59 06 20
0160  ED 78 7D D9 5F D9 7C D9 57 D9 D9 79 C6 40 4F 30
0170  09 C6 40 4F ED 5F EE 80 ED 4F D9 C3 D1 00 ED 5F
0180  08 54 26 C0 D9 78 D9 FE 03 20 0E 06 08 7A ED 4F
0190  7E 17 CB 13 2C 10 F6 18 34 FE 02 20 11 06 04 7A
01A0  ED 4F 7E 17 CB 13 17 CB 13 2C 10 F3 18 1F FE 01
01B0  20 17 7A ED 4F 7E 2C 07 07 07 07 E6 F0 5F 7A ED
01C0  4F 7E 2C E6 0F B3 5F 18 04 7A ED 4F 7E 08 ED 4F
01D0  7D FE 01 7A CE 00 67 FD E9 ED 5F 08 54 26 C0 D9
01E0  78 D9 FE 03 20 0D 06 08 7A ED 4F 73 CB 03 2C 10
01F0  F7 18 2C FE 02 20 0F 06 04 7A ED 4F 73 CB 03 CB
0200  03 2C 10 F5 18 19 FE 01 20 11 7A ED 4F 73 2C 7B
0210  0F 0F 0F 0F 7A ED 4F 77 2C 18 04 7A ED 4F 73 08
0220  ED 4F 7D FE 01 7A CE 00 67 FD E9 0D 2A 2A 2A 20
0230  50 52 49 4E 54 45 52 20 42 55 46 46 45 52 20 2A
0240  2A 2A 0D 0A 20 28 43 29 20 31 39 38 34 20 20 52
0250  2E 47 2E 48 61 72 76 65 79 0D 0A 0A 00 20 38 31
0260  39 32 00 31 36 33 38 34 00 33 32 37 36 38 00 36
0270  35 35 33 36 00 20 42 59 54 45 53 20 4F 46 20 52
0280  41 4D 20 49 4E 53 54 41 4C 4C 45 44 0D 0A 00 21
0290  21 21 20 52 4F 4D 20 43 48 45 43 4B 53 55 4D 20
02A0  45 52 52 4F 52 20 21 21 21 0D 0A 00 A3 FF FF FF
```

test facility which is enabled by disconnecting link 1. The effect of this is that whenever the buffer is powered up, a signon message is inserted into the buffer memory giving the size of DRAM present. This will be printed on the connected printer. The message is only printed once and will not appear on subsequent buffer resets using the reset button.

As a further check to the integrity of the system the EPROM contains code to check itself and will indicate on any connected printer if there appearts to be an error in the EPROMs code. This will also help to diagnose any problems with programming the EPROM.

Construction

Despite the fact that the workings of the hardware are complex, the physical construction of the circuit is quite straightforward especially if a PCB is used. Having said this it should be borne in mind that every chip should have a decoupling capacitor connected directly across its power supply pins, and this is even more important for the DRAM chips which may suffer from RAM errors if this is not done. The DRAM does not need to be very fast, 450nS access time is adequate, because the access cycle time generated by the circuit is quite long. Finally, the input and output ports from the circuit should be connected by short lengths of cable to Centronics connectors one plug and one socket.

"... the design features a minimum of hardware and very little scratchpad memory ..."

Conclusion

Although the main purpose of this article was to describe the printer buffer, I hope it has given a little insight into the design of standalone microprocessor based devices. The challenge of designing a system with the minimum of hardware and very little scratchpad memory is, to my mind, far more rewarding than designing purely random logic based devices or totally software based projects.

The software for this project is available in EPROM from the author at the all inclusive price of £6.50. Please send cheques/POs to: *R. Harvey, Buffer Eprom, 30 Jericho St., Oxford, OX2 6BU.*

READERS WISHING TO OBTAIN A PCB FOR THIS PROJECT SHOULD PHONE 01-833-0846.

PARTS LIST

Semiconductors	
IC1	Z80
IC2	2716
IC3	8255
IC4	74LS00
IC5	74LS06
IC6-7	4164
IC14	7805A
Resistors	
R1	10k
R2, R3	820R
R4-7	4k7
Capacitors	
C1	100u 25V
C2	4n7
C3	22u
CD x 14	1n disc ceramic

SINCLAIR SECTOR

Sinclair Sector is a new E&CM regular to keep you up to date with the latest hardware and software developments for ZX Spectrums and ZX81s. Compiled by Stephen Adams.

Sinclair's world still seems to be dominated by what has happened to the QL, some software houses seem to have some prototypes, but are being upset by the fact that the QDOS operating system is not yet finished. There also seem to be problems with the ULAs that Sinclair is using to store most of the circuitry, they seem to have either design errors in them or are unreliable in production quantities.

Hardware manufacturers are biting at the leash by announcing products that are QL compatible, like the RS232 to Centronics converter from Miracle Products and the Disc drive system from Xcom. The Xcom has an advantage though as it already works off any RS232 port (the way of putting the software into the machine being the problem here). Advertising a product that either does not exist or has yet to be finished seems to be one of the common things among both hardware and software manufacturers alike.

Review guidelines

I, like any other reviewer SHOULD do, like to see a product that is ready for the customer to use before testing it. But recently I have been trying to test two bits of equipment that, although they have been on sale for some time, are still not finished.

Some of the faults included hardware problems like printed circuit boards which moved about in the box, boards which contained standard sockets with non standard connections (and with no warnings in the instructions), and equipment which worked OK if you demonstrated it for ½ hour (any longer than an hour and it crashed!).

Software problems never seem to show up until the reviewer points them out (too late for the poor customer), and they usually take longer to sort out as neither reviewer or manufacturer has any idea what they are looking for.

So **please** manufacturers don't send in equipment for review that is not finished (or looks like a birds nest inside) and do supply us with some software which allows us to try out the various facilities without writing entire programs (preferably listed in the manual so that the user can use them too). This saves valuable time on both sides and allows the user to buy reliably tested equipment and the manufacturer to get good reviews of his product.

Modem matters

Using modems seems to be the "in thing" at the moment. Although to be legal you need to have a PO jack socket and use British Telecom approved equipment connected to a telephone, this has not stopped many people taking up this new hobby.

Many bullitin boards as they are called exist run by individuals as well as companies and are free to use. All that is required is a 300 baud modem (in some cases 1200/75 baud models need to be used for databases like PRESTEL), a RS232 interface which is hardware controlled (unlike Sinclair's Interface 1) and a terminal program. Apart from the modem these can be bought quite cheaply for about £50.

The modems for 300 baud can be built by the user (as in the case of the popular Maplin modem) for about £50, but these still need approval which will cost you a minimum of £5,000 as well as a long time to get approval. Modems which are legal are the Buzz box from Scicon for 300 baud (£115) and for the 1200/75 type the Prism modem 2000 (£99).

Second hand modems can also be bought from Display Electronics.

Depending on the software you can download software over the telephone line from these boards as well as being able to keep up with the latest gossip.

A ZX81 interface and software is also available for those with a 16K RAM pack from Microcomputer Resources at £29.95 for 1200/75 or 300 baud with 40 characters per line (PRESTEL style)! This adaptor only features black and white and is not usable on ASCII based Bullitin boards.

Spectrum standards

Software for the Spectrum seems to be settling down with some standards appearing for business software.

The Tasword 2 (from Tasman Software) wordprocessor at up to 64 characters per line and approx 300 lines per file seems to be popular for the 48K Spectrum, as does the Masterfile program from Campbell Systems.

Both of these can be modified for use with microdrives and 80 column printers. Campbells latest version also supports up to 51 characters per line.

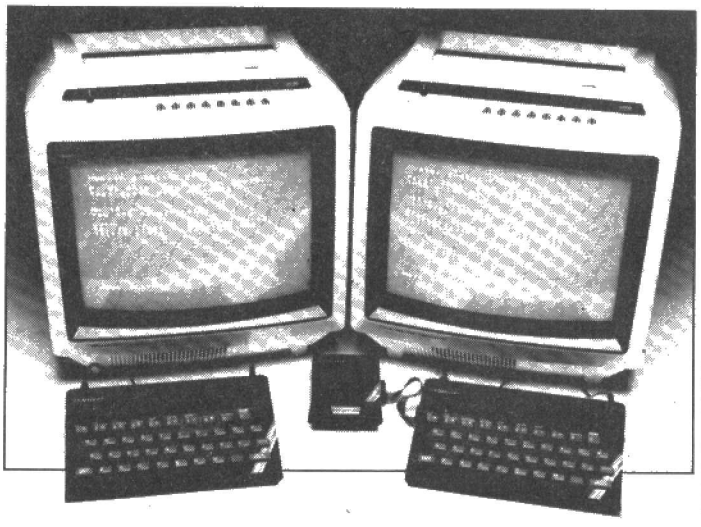
As far as electronics software is concerned a firm called Spectre seem to have the right idea, ELECTRONICS a program that allows you to construct a circuit made up of various components. The computer will then trace through all the branches detailing whether a connection is on or off from a particular set of input conditions. It is slow as it is entirely written in BASIC, but the hi-res drawing of resistors, capacitors and transistors make up for this, as the diagram can be COPY'd. Their latest program is an analogue version of the first (LINNET) allows you to set voltages and have the computer work out the result at various points.

Sinclair has now appointed a distributor for its out of guarantee spares. Combined Precision Components (not to be confused with a cable manufacturer with the same initials) can supply all the components for a ZX81 or ZX Spectrum, Sinclair's RAM packs and printers. It can also supply a Service Manual for the Spectrum (£15) and the ZX printer (£10), which is one thing Sinclair has kept very much under cover. So if you intend to repair your Sinclair yourself contect them for spare parts. It must be pointed out that these are trade distributors and therefore are not an advice service for problems with Sinclair machines!

A good Sinclair quote was from one of the BBC computer magazines about why BBC BASIC II was commissioned, it said "This was because the ZX81 was found to have better mathematics than BASIC 1".

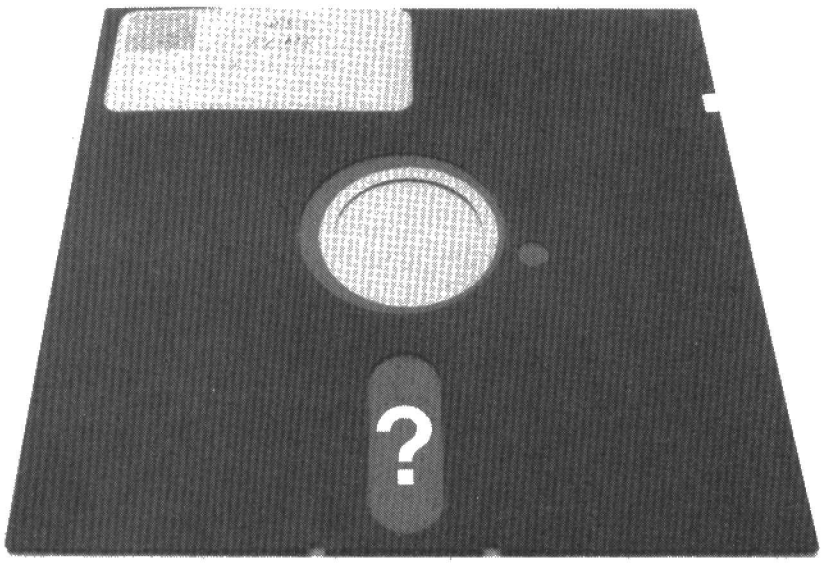
Another electronics magazine has published a project to "freeze frame" your micro (to watch it step through programs etc). The only problem is that it will not work on dynamic RAM machines. The only ones that I know that it will work on is the BBC or 1K ZX81's! The BBC however uses interrupts which are time dependent and the ZX81 uses the refresh cycle (which does not work) to put out the screen!

Send any ZX news to Stephen Adams c/o E&CM 155 Farringdon Road, London EC1R 3AD.



Combined Precision Components Ltd., 194-200 North Road, Tel: 0772-555034.
Campbell Systems, 15 Rous Road, Buckhurst Hill, Essex IG9 6BL.
Spectre Software, Enfield House, Swardeston, Norwich, Norfolk.
Miracle Systems Ltd., 6 Armitage Way, Kings Hedges, Cambridge CB4 2UE.
Prism Computer Products Ltd., Prism House, 18-29 Mora Street, City Road, London EC1 8BT.
Tasman Software, 17 Hartley Crescent, Leeds LS6 2LL.
XCOM (Services) Ltd., 640 High Road, London E10 6RN. Tel: 01-539-4147.
Display Electronics, 32 Biggin Way, Upper Norwood, London SE9 3XF. Tel: 01-679-4414.
Scicon, Brick Close, Kiln Farm, Milton Keynes, MK11 3E. Tel: 0908-567567.
Maplin Electronics, PO Box 3, Rayleigh, Essex S86 8LR.
Microcomputer Resources, 1 Branch Road, Park Street Village, St. Albans, Herts. Tel: 0727-72917.

Lost sector



The advantages of disk drives are obvious, but a hardware breakdown can be a disaster. If a file chain breaks part of the file will be lost forever – but T. E. Edward's 'Lost Sector' can find it for you.

Using disk drives is almost mandatory nowadays if you want to run a business computer system, however small. More and more hobbyists are also taking the financial plunge and investing in disks. The advantages are obvious especially if you are blessed with a really good Disk Operating System like FLEX. However, the price for that extra speed and easy file handling is complexity. Hardware failures, if not prepared for, can be very expensive in time and money. I regularly back-up both my system and work disks so that when (not if!) something gets corrupted it is usually an easy job to recover. Being human though, I occasionally forget to do the back-up for some time and that well known law immediately comes into effect! The disk repair utility programs supplied with FLEX are for the most part very good and the explanation of the disk file structure is extremely good. There is however one particular problem that comes up every so often which can cause a great deal of trouble.

When a file chain on the disk gets broken the part after the break is entirely irrecoverable; so that after the files have been recovered as much as possible there will still be some sectors 'lost' on the disk. With 1140 sectors on an 8" and 790 sectors on a 40 track disk it does not matter too much if two or three sectors are not usable. But if a big file or the 'free chain' becomes broken then the 'lost' sectors can easily run into two or three hundreds. I have even had cases where all the free space on the disk has disappeared! The way of recovering these sectors is to find the beginning of the lost part of the chain and tack it on to the end of (usually) the free chain. Then the end pointer and sector count in the System Information Record in sector 3 has to be adjusted with one of the disk repair utilities like EXAMINE or DISKEDIT.

The real trouble starts when you try to find the start of the 'lost' chain. Nothing points to it! By the very nature of the failure the chain is isolated. To reduce the space overhead in each sector FLEX puts only a forward pointer in to link the chains and this is the very pointer that gets corrupted. (For some reason when the corruption takes place the pointer is usually made zero, which makes it look like the end of the chain).

The very fact that the head of the chain is isolated can be used to find it. THE FIRST SECTOR OF A LOST CHAIN HAS NO OTHER SECTOR POINTING AT IT! So all we have to do now is make a two dimensional matrix of 'tracks per disk' x 'sectors per track' and then read the disk sequentially, setting those elements in the matrix that we find pointers pointing to. When the disk has been entirely read, every element should be set to show that every sector is pointed to by another sector. Now it is just a matter of scanning the matrix to make sure every bit is set. Any bits that are not set have their position reported as a track-sector number. These sectors are the head of 'lost' chains.

The utility program I have included here to find lost sectors is by no means elegant as I am an engineer, not a software writer. It fits comfortably in the 'utility' area of FLEX itself but can easily be relocated by changing the origin and re-assembling it.

The program is divided into four basic parts:-

INITALISE: Lines 63 to 107 set Flex parameters, get the drive number, read basic information from the SIR sector of the disk, and clear out the matrix area. This section is fairly straightforward except perhaps for the way the SIR is read. I could have simply used a 'read sector' FMS function but instead decided to use the TSC approved method. This I discovered by looking through one or two or their utility programs.

FILL: From lines 111 to 198 the program searches the disk using the track-sector sizes previously read from the SIR record. The free chain and directory are dealt with first, followed by the rest of the disk. Before an element in the matrix is set, a check is made to see if it is already set (ie has been pointed to by another sector). If this is the case then a 'Chain Conflict' error message is printed and the program carries on. At any time while reading the disk an 'abort' can be performed by typing an 'E'.

PRINT: Between lines 258 and 284 the matrix is scanned to make sure that every bit is set. Error messages are generated for those that are not. The first four sectors of track zero are skipped as they are special and will not be part of any chain.

CLOSE: Tidying up is performed between lines 289 and 297 and is straightforward.

The program can handle any mix of Flex disk format up to 256 tracks of 256 sectors and so is suitable for all floppies and 'hard' disks (including Winchesters) up to that size. (That is Flex's maximum as well).

Although the program itself runs in the utility area, it uses memory from $0100 upwards for the element matrix. The amount required is directly proportional to the size of the disk. 'Packed' format is used so that each sector is represented by one bit; to find how much memory your disk will need, apply the following formula: number of tracks multiplied by number of sectors divided by eight. It would have been a lot easier to have used one byte per sector but for a maximum size disk that would have needed 64K bytes just for the matrix!

The author is prepared to put the source and object on to 5" or 8" reader supplied floppy disks if return postage is included. The address is:- *The Old Forge, Little St. Mary's, Long Melford, Suffolk, CO10 9HX.*

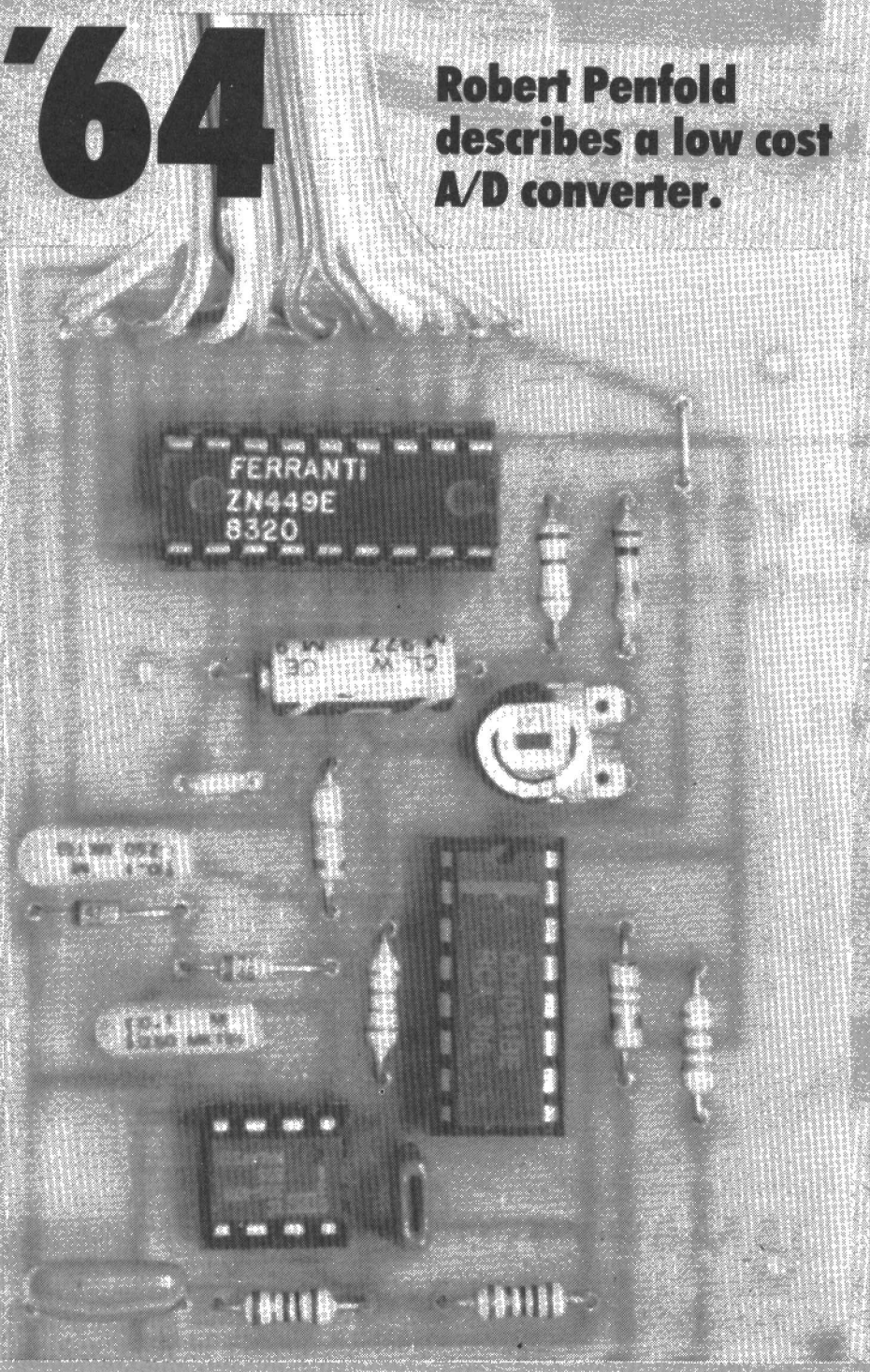
Lost Sector: Finds sectors not pointed to

```
                        ********** EQUATES **********
                GETCHR  EQU   $CD15
                PUTCHR  EQU   $CD18
                PSTRNG  EQU   $CD1E
                CRLF    EQU   $CD24
                INBUFF  EQU   $CD1B
                RPTERR  EQU   $CD3F
                GETHEX  EQU   $CD42
                OUTADR  EQU   $CD45
                STAT    EQU   $CD4E
                FMS     EQU   $D406
                FCB     EQU   $CD4C
                FMSCLS  EQU   $D403
                WARMS   EQU   $CD03
                TTYWID  EQU   $CC04
                MATRIX  EQU   $0100     START OF MATRIX
                WIDTH   EQU   80        Set to screen width
                *
0100            MATRIX  ORG   $0100     <-
                *
                ********** CONSTANTS **********
                *
0100            RUNTRK  RMB   1         Max track of current disk
0101            RUNSEC  RMB   1         Max sec/trk of current disk
0102            NOWTRK  RMB   1         Track pointer
0103            NOWSEC  RMB   1         Sector pointer
0104            WORK    RMB   2         SCRATCH
0106            REM     RMB   1         Remainder
0107            FLAG    RMB   1
0108            SAVWID  RMB   1
0109 01 02 04 08 LOOKUP  FCB   $01,$02,$04,$08,$10,$20,$40,$80
                *
                ********** MAIN PROG **********
                *
C111 BD  0B     START   BSR   INIT      INITALISE
C113 BD  70             BSR   FILL      Fill matrix
C115 17  0198           LBSR  PRINT     Print lost sectors
C118 17  0183           LBSR  CLOSE     TIDY UP
C11B 7E  CD03           JMP   WARMS
                *
                *
                ********** INITALISE **********
                *
C11E B6  CC04   INIT    LDA   TTYWID    Save FLEX width
C121 B7  C108           STA   SAVWID
C124 B6  50             LDA   #WIDTH    Set new width
C126 B7  CC04           STA   TTYWID
C129 8E  C332           LDX   #MSG3
C12C BD  CD1E           JSR   PSTRNG
C12F BD  CD1B           JSR   INBUFF    Get drive no.
C132 BD  CD42           JSR   GETHEX
C135 2A  09             BCC   INIT2     OK...
C137 8E  C306   INIT1   LDX   #MSG1
C13A BD  CD1E           JSR   PSTRNG
C13D 16  01AE           LBRA  CLOSE
                *
C140 1F  10     INIT2   TFR   X,D
C142 C1  00             CMPB  #$0       Less than 0 ?
C144 2D  F1             BLT   INIT1     YES...
C146 C1  04             CMPB  #$4       More than 4 ?
C148 2E  EF             BGT   INIT1     YES...
                *
C14A 8E  C35C           LDX   #MSG5
C14D BD  CD1E           JSR   PSTRNG
C150 8E  CB40           LDX   #FCB
C153 E7  03             STB   3,X       Drive number
C155 B6  10             LDA   #$10
C157 A7  84             STA   0,X       Read DIR
C159 BD  D406           JSR   FMS
C15C 27  06             BEQ   INIT3     NO ERR...
C15E BD  CD3F   INIT21  JSR   RPTERR    Report
C161 16  0198           LBRA  CLOSE
                *
C164 B6  07     INIT3   LDA   #7         Get record code
C166 A7  84             STA   0,X       Put in FCB
C168 BD  D406           JSR   FMS
C16B 26  F1             BNE   INIT21    ERR...
C16D 30  88 66          LEAX  64+38,X   Point to data
C170 BD  CD45           JSR   OUTADR    Print disk's tk-sec
C173 AE  1F             LDX   -1,X       tk-sec to X
C175 BF  C100           STX   RUNTRK
                *CLEAR MATRIX
C178 8E  0100           LDX   #MATRIX
C17B B6  00             LDA   #0
C17D A7  80     CLRM    STA   0,X+
C17F 8C  2100           CMPX  #MATRIX+$2000 End of matrix ?
C182 26  F9             BNE   CLRM      Again...
C184 39                 RTS
                *
                ********** FILL MATRIX **********
                *
C185 8E  CB40   FILL    LDX   #FCB      Get pointer to free chain
C188 EE  88 5D          LDU   64+29,X
C18B 17  00AB           LBSR  MAP       Process it
                *
C18E B6  06             LDA   #$6       Open directory code
C190 A7  84             STA   0,X
C192 BD  D406           JSR   FMS
C195 26  C7             BNE   INIT21    Error...
C197 B6  07             LDA   #$7       Get inf. rec. code
C199 A7  84             STA   0,X
C19B BD  D406   DIR     JSR   FMS
C19E 27  08             BEQ   DIR1      No error...
C1A0 A6  01             LDA   1,X       Get error code
C1A2 81  0B             CMPA  #$0B      EOF ?
C1A4 26  BB             BNE   INIT21    Aborts...
C1A6 20  0E             BRA   FILL10    Finished directory
                *
C1A8 A6  04     DIR1    LDA   4,X       Get 1st chars
C1AA 2B  EF             BMI   DIR       Deleted entry
C1AC 27  ED             BEQ   DIR       Unused  entry
C1AE EE  88 11          LDU   17,X      Get tk-sec for this file
C1B1 17  0082           LBSR  MAP       Process it
C1B4 20  E5             BRA   DIR       Get another directory entry
                *
C1B6 7F  C102   FILL10  CLR   NOWTRK    Reset track pointer
C1B9 B6  04             LDA   #4        (Jump boot sectors)
C1BB B7  C103           STA   NOWSEC    Reset sector pointer
                *
C1BE BD  CD4E   FILL1   JSR   STAT      Break off ?
C1C1 27  0D             BEQ   FILL11
C1C3 BD  CD15           JSR   GETCHR
C1C6 81  45             CMPA  #'E
C1C8 26  06             BNE   FILL1!
C1CA 17  0121           LBSR  CLOSE
C1CD 7E  CD03           JMP   WARMS
C1D0 8E  CB40   FILL11  LDX   #FCB
C1D3 10BE C102          LDY   NOWTRK    Get current tk-sec
C1D7 10AF 88 1E         STY   30,X      Put in FCB
C1DB 86  09             LDA   #9        Read sector code
C1DD A7  84             STA   0,X       Put in FCB
C1DF BD  D406           JSR   FMS       Read it !
C1E2 27  96             BEQ   FILL2
C1E4 BD  CD3F           JSR   RPTERR
C1E7 16  0104           LBRA  CLOSE
C1EA EE  88 40  FILL2   LDU   64,X      Read pointer
C1EE 11B3 C100          CMPU  RUNTRK    Valid pointer ?
C1F1 2F  0E             BLE   FILL3     Yes...
C1F3 8E  C344           LDX   #MSG4
C1F6 BD  CD1E           JSR   PSTRNG
C1F9 8E  C102           LDX   #NOWTRK
C1FC BD  CD45           JSR   OUTADR    Print bad pointer
C1FF 20  02             BRA   FILL7     Abort this sector
                *
C201 BD  33     FILL23  BSR   MAP
C203 B6  C103   FILL7   LDA   NOWSEC    Finished this tk ?
C206 B1  C101           CMPA  RUNSEC
C209 26  1C             BNE   FILL5     No...
C20B B6  C102           LDA   NOWTRK    Finished disk ?
C20E B1  C100           CMPA  RUNTRK
C211 26  19             BNE   FILLA     No...
                *
C213 39                 RTS             Exit 'FILL'
                *
C214 34  10     CELICT  PSHS  X
C216 FF  C104           STU   WORK      Store pointer
C219 8E  C100           LDX   #MSG2
C21C BD  CD1E           JSR   PSTRNG
C21F 8E  C104           LDX   #WORK     Point to pointer
C222 BD  CD45           JSR   OUTADR    Print it
C225 35  90             PULS  X,PC
                *
C227 7C  C103   FILL5   INC   NOWSEC    Update sector pointer
C22A 20  92             BRA   FILL1
                *
C22C 7C  C102   FILL6   INC   NOWTRK    Update track pointer
C22F 86  01             LDA   #1
C231 B7  C103           STA   NOWSEC    Reset sector pointer
C234 20  88             BRA   FILL1
                *
                *
C236 1183 0000  MAP     CMPU  #0        End of chain ?
C23A 26  01             BNE   MAP1      No...
C23C 39                 RTS
C23D 1C  FE     MAP1    CLC
C23F BD  06             BSR   ELEM     Clear ?
C241 24  02             BCC   MAP2      No conflict...
C243 8D  CF             BSR   CFLICT    Print conflict
C245 1A  01             SEC             Set it
C247                    *
                        ********************************
                        * ELEMENT SUBROUTINE
                        *Entered with TK/SEC in 'U'.
                        *If carry set then sets element.
                        *If carry clear then reads element
                        *into carry.
                        *
                C247    ELEM    EQU   *
C247 34  17             PSHS  A,B,X,CC
C249 1F  30             TFR   U,D
C24B F7  C105           STB   WORK+1    Save sector
C24E 7A  C105           DEC   WORK+1    Adjust sector numbering
C251 7F  C104           CLR   WORK      Clear MS byte
C254 F6  C101           LDB   RUNSEC    Get multiplicand
C257 3D                 MUL             Multiply
C258 F3  C104           ADDD  WORK      Add in sectors
                        *'D' Now has absolute sector number
                        *Now divide by eight
C25B FD  C104           STD   WORK
C25E 86  01             LDA   #1        Set count
C260 7F  C106           CLR   REM
C263 7A  C105   ELEM1   LSR   WORK+1    L.S. byte
C266 24  06             BCC   ELEM2     No remainder...
C268 1F  89             TFR   A,B
C26A FB  C106           ADDB  REM
C26D F7  C106           STB   REM
C270 74  C104   ELEM2   LSR   WORK      M.S. byte
C273 24  08             BCC   ELEM3     No remainder
C275 C6  80             LDB   #$80      Set M.S. bit
C277 FA  C105           ORB   WORK+1
C27A F7  C105           STB   WORK+1
C27D 48         ELEM3   LSLA            Shift count up
C27E B1  0B             CMPA  #8
C280 2A  E1             BNE   ELEM1     Not finished...
C282 FC  C104           LDD   WORK
                *
C285 C3  0100           ADDD  #MATRIX   Set base address
C288 1F  01             TFR   D,X       Mat byte pointer
C28A A6  84             LDA   0,X       Get byte
C28C 10BE C109          LDY   #LOOKUP   Set base
C290 FA  C106           LDB   REM
C293 E6  A5             LDB   B,Y       Get correct mask
C295 F7  C106           STB   REM       Mask to RAM
C298 27  0C             BEQ   ELEM4     Was 'C' set
C29A 25  0D             BCS   ELEM4     Set bit
C29C 1C  FE             CLC             Clr carry
C29E 84  C106           ANDA  REM
C2A1 26  02             BNE   EL1       Bit set
C2A3 20  09             BRA   ELEM5
C2A5 1A  01     EL1     SEC
C2A7 20  05             BRA   ELEM5
C2A9 BA  C106   ELEM4   ORA   REM
C2AC A7  84             STA   0,X       Store to array
C2AE 35  94     ELEM5   PULS  B,A,X,PC
                *
                *
                ********** PRINT LOST SECTORS **********
                *
C2B0 7F  C107   PRINT   CLR   FLAG
C2B3 86  FE             LDA   #$FF      Set to -1
C2B5 CA  04             LDB   #$4       Skip special sectors
C2B7 4C         PR1     INCA
C2B8 5C         PR2     INCB
C2B9 1F  03             TFR   D,U
C2BB 34  02             PSHS  A
C2BD 1C  FE             CLC             Read element
C2BF 8D  B6             BSR   ELEM      Is it set ?
C2C1 25  1E             BCS   PR4       Yes...
C2C3 7D  C107           TST   FLAG      Print heading ?
C2C6 26  09             BNE   PR3       No...
C2C8 8E  C383           LDX   #MSG6
C2CB BD  CD1E           JSR   PSTRNG
C2CE 7C  C107           INC   FLAG
C2D1 FE  C104   PR3     STU   WORK
C2D4 8E  C104           LDX   #WORK
C2D7 BD  CD45           JSR   OUTADR    Print bad sector
C2DA 20  02             LDA   #$20
C2DC BD  CD18           JSR   PUTCHR
C2DF 35  02     PR4     PULS  A
C2E1 F1  C101           CMPB  RUNSEC    End of track
C2E4 26  02             BNE   PR2       Another sector...
C2E6 C0  00             LDB   #$0
C2E8 B1  C100           CMPA  RUNTRK    End of disk ?
C2EB 26  CA             BNE   PR1       Another track...
C2ED 39                 RTS
                *
                *
                *
                ********** CLOSE DOWN & TIDY UP **********
                *
C2EE BD  CD24   CLOSE   JSR   CRLF
C2F1 7D  C107           TST   FLAG
C2F4 26  06             BNE   CLS1      Errors found
C2F6 8E  C384           LDX   #MSG7
C2F9 BD  CD1E           JSR   PSTRNG
C2FC B6  C108   CLS1    LDA   SAVWID    Restore FLEX width
C2FF B7  CC04           STA   TTYWID
C302 BD  D403           JSR   FMSCLS
C305 39                 RTS
                *
                *
                *
                *
C306 0D 0A 49 4E MSG1    FCC   $0D,$0A,/Invalid drive number ;/,$0A,$0D,4
C320 43 4F 41 6E MSG2    FCC   /Chain conflit at ;/,4
C332 0A 0D 44 72 MSG3    FCC   $0A,$0D,/Drive number ? /,4
C344 0D 0A 42 41 MSG4    FCC   $0D,$0A,/Bad chain pointer at ;/,4
C35C 0D 0A 54 72 MSG5    FCC   $0D,$0A,/Track-sector count for this disk is ;/,4
C383 0D 0A 46 69 MSG6    FCC   $0D,$0A,/First sector of lost chains: /,4
C3A4 0D 0A 4E 6F MSG7    FCC   $0D,$0A,/No lost chains found/,4
                *
                         END   START
```

Commodore 64 A/D Add-on

SOUPING UP THE '64

Robert Penfold describes a low cost A/D converter.



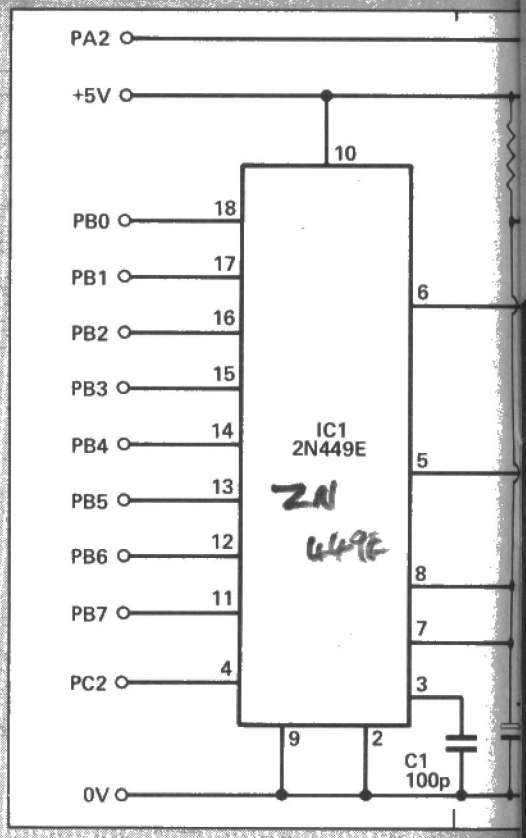


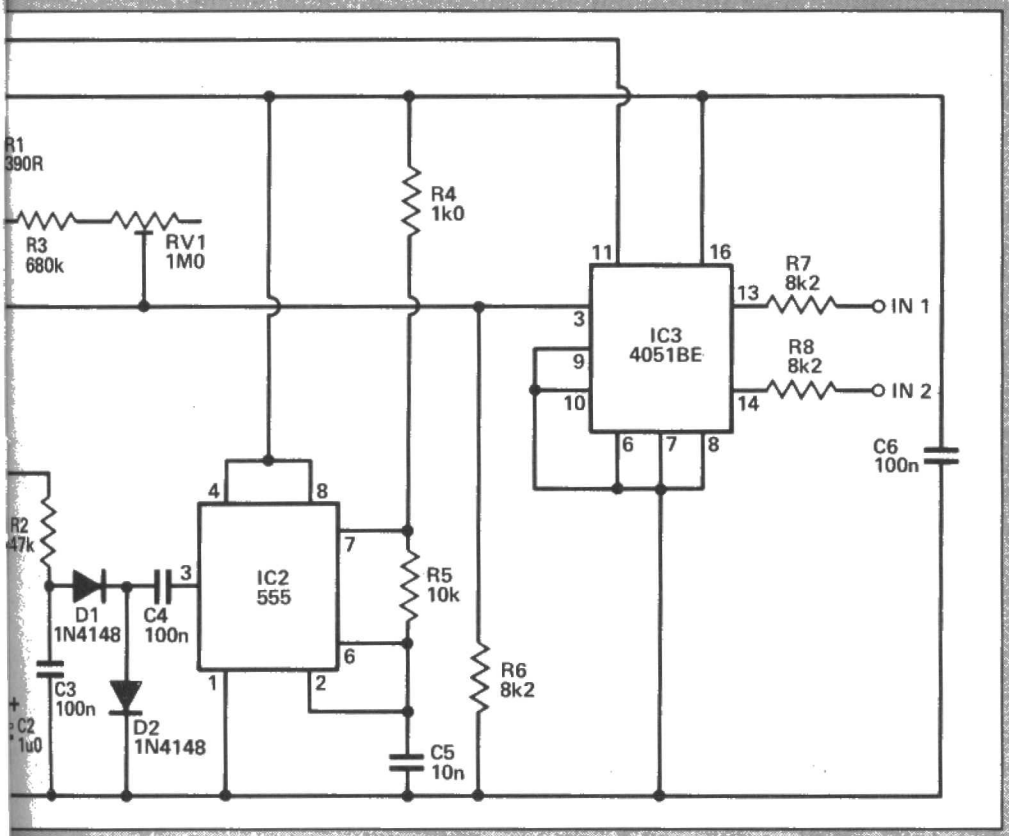
Figure 2. Full circuit diagram.

For a computer in its price range the Commodore 64 has excellent interfacing potential, and a useful range of built-in interfaces. The user port is probably of most interest to the computer add-on enthusiast, although the cartridge port gives potential for further expansion. The user port has the advantage of taking a standard 2 x 12-way 0.156 inch female edge connector so that connections can be made to it quite easily, and it provides an 8 bit input/output port plus three handshake lines. The CBM64 can therefore be connected to the outside world via the user port with little or no extra hardware, depending upon the particular application concerned.

One obvious omission from the built-in interfaces is an analogue to digital converter. There are actually two analogue inputs at each joystick port, but these are designed to directly respond to the resistance of the potentiometer in a games paddle, and are not the more usual voltage sensitive type. These could be used in some applications, but in most cases a voltage sensitive converter is required. Fortunately, an analogue to digital converter can be added to the user port with little difficulty, and the design featured here is a two channel, 8 bit type, which has a maximum conversion rate of over 10 0000 per second. Although designed specifically for use with the CBM64 it should in fact be usable with any other computer which has a similar port (such as the VIC-20, PET, and BBC model B).

Basic arrangement

As mentioned above, the user port of the CBM64 has an 8 bit input/output port, and



does not vary significantly (nine clock cycles). A timing loop can therefore be used to prevent the computer from taking a premature reading. However, the status output is made available to the user on the printer circuit board so that it can be used if desired.

An electronic switch at the input of the converter enables one of two input signals to be connected through to the converter. The required input is selected by setting up PA2 as an output and then placing this at the appropriate logic level.

The converter requires a negative supply voltage which is not available from the CBM64 user port. This supply could be generated from one of the 9 volt AC outputs of the user port, but the alternative method used here is to use an AF oscillator plus a rectifier and smoothing circuit.

The circuit

A ZN449 analogue to digital converter is used in this design, as can be seen by referring to the circuit diagram of **Figure 2.**

The ZN449 is a relatively recent device from Ferranti, and it is very convenient in use. It has a built-in clock oscillator which only requires a single discrete component (C1) which sets the operating frequency.

each of these lines is individually programmable as an input or an output. There are three handshake lines, one of which is PC2. This is an output which pulses low for about 1uS after each read or write operation to the 8 bit port. FLAG is a handshake input, and this is a negative edge sensitive type. In other words, one bit of an internal register of the 6526 interface device utilized for the user port is set as the FLAG input is taken from the high to low. The third handshake line (PA2) is actually a line from port A of the 6526 interface device, and it can be programmed to operate as an ordinary input or an input, as required.

Figure 1 shows the system used in this A/D converter. The 8 bit port (PB0 to PB7) is used with all lines as inputs, and these take the 8 bit output of the converter. A brief negative "start conversion" pulse is required by the converter each time a reading is to be taken, and this is provided by PC2. This is a convenient way of doing

"An obvious omission from the built in interfaces is an analogue to digital converter . . ."

things if a rapid series of readings are to be taken, since a "start conversion" pulse is provided by PC2 immediately after each reading of the converter is taken. If readings are taken only infrequently this system operates slightly less well, since the input voltage will be read by the converter as soon as its output has been read by the computer, but this new reading will not be

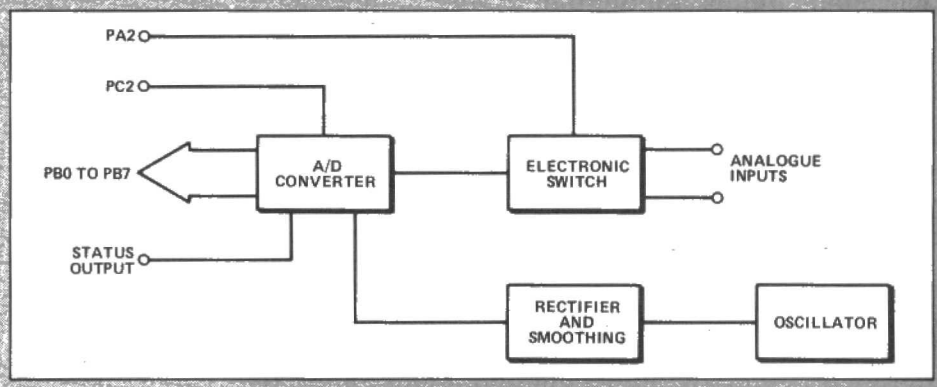


Figure 1. Block diagram of the A/D converter. The oscillator section is necessary to generate the negative voltage required by the A/D IC.

transferred to the computer for some time. In many applications the fact that the readings passed to the computer are not current will be of no consequence, but for real-time applications there is an easy way around the problem anyway. POKEing any number to the 8 bit port will cause a new conversion to be undertaken so that fresh data can then be read, but this will not have any detrimental effect, bearing in mind that the 8 lines of the port are all set as inputs.

The converter has a status output which goes low while a conversion is in progress. When readings must be taken as rapidly as possible this output can be monitored by the computer, with readings being taken after each low to high transition of the status output is detected. The FLAG input of the CBM64's user port is negative rather than positive edge sensitive, and cannot be used to directly monitor the status output properly. An inverter could be used to give a signal of the correct polarity for the FLAG input, but this is not really necessary since the conversion time of the circuit

The specified value of 100p for C1 gives a nominal operating frequency of just over 900kH, which is close to the guaranteed maximum usable clock frequency for the ZN449 of 1MHz. In practice this capacitor can invariably be made somewhat lower in value to give a faster conversion time in applications where a high conversion rate is needed. The ZN449 uses the successive approximation conversion technique, and even using the specified value for C1 more than 100 000 conversions per second can be achieved. Incidentally, the ZN449 has a accuracy of 1 LSB, but for critical applications the more accurate ZN448 (1/2 LSB) or ZN447 (1/4 LSB) can be used.

The ZN449 has an integral precision (2.55 volt) reference source, but this needs discrete load resistor R1 and decoupling capacitor C2. The outputs of the device are tri-state types, but as in this case it is not being directly interfaced to the computer's data bus the outputs are permanently enabled by taking pin 2 of IC1 to the negative supply rail.

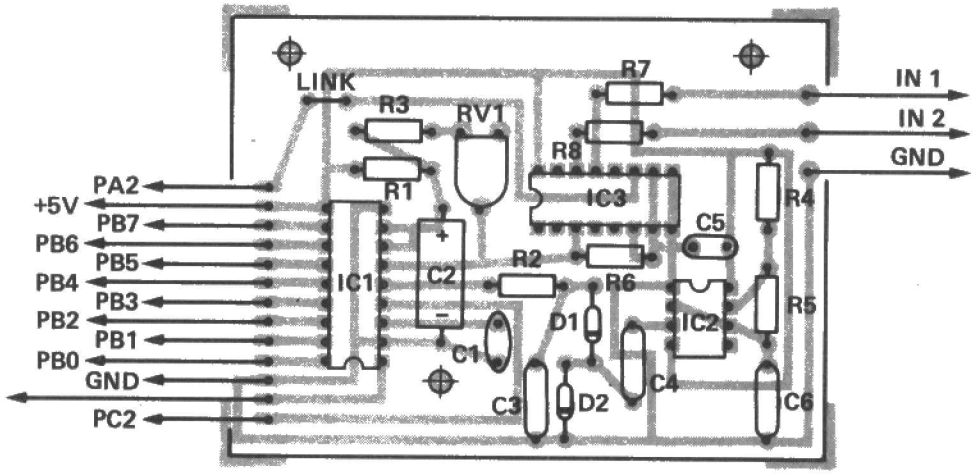
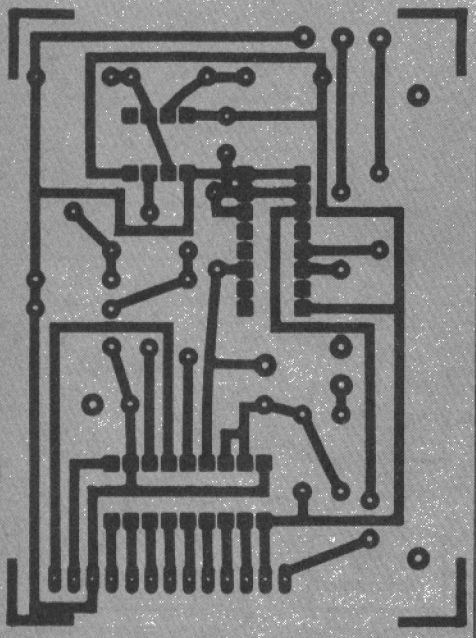


Figure 3a (above) shows the A/D converter's overlay while Figure 3b (below) shows the foil pattern.



A negative supply of between 3 and 30 volts is needed for the "tail" resistor of IC1. This resistor (R2) is part of the voltage comparator circuit, and the negative supply enables input voltages down to the negative supply voltage to be measured. IC2 is used in the standard 555 astable mode, and its output is rectified and smoothed by D1, D2, and C3 to give a potential of approximately –3 volts. Little output current is available here, but the current drain through R2 is only about 65 microamps.

IC3 is the analogue switch, and this is a CMOS 4051 eight input analogue switch. In this case there is only one output available to drive the device, and so two of the control inputs and six of the signal inputs are unused. Nevertheless, this device still represents a convenient and economic way of providing a second input. Of course, only one input signal at a time can be converted, and with two inputs in use the maximum conversion rate is only around 50 000 per second (which is still more than adequate for the vast majority of uses). R6 to R8 set the input sensitivity of the circuit at approximately 0 to 5 volts, but this can be changed to practically any desired range by an attenuator or an amplifier added at the input. VR1 is the zero adjustment control.

Construction

The printed circuit design for this project appears in **Figure 3.** As IC1 is a fairly expensive device and IC3 is a CMOS type it is strongly recommended that IC sockets should be used for these. Be careful to fit the integrated circuits the right way round, especially IC3. Do not overlook the single link wire near R3.

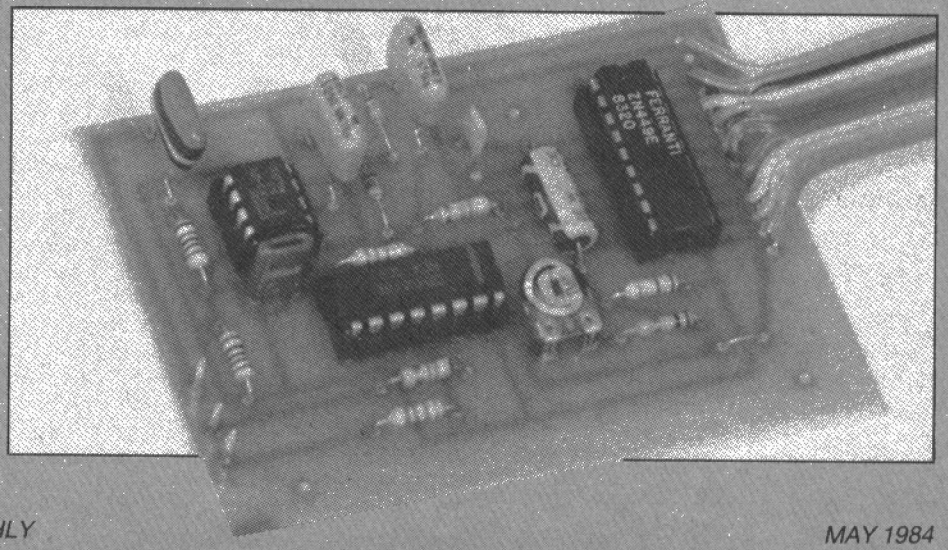
A piece of 10 way ribbon cable aboput 0.5 to 1 metre long is used to connect the board to the user port of the CBM64. One end of the cable is connected to the board, and this is not too difficult provided a small amount of insulation is stripped from the end of each lead, and they are tinned with solder prior to connecting the cable. It is not advisable to make these connections to the board via pins as the close spacing of the connections would make accidental short circuits difficult to avoid. A 2 by 12 way 0.156 inch edge connector is fitted to the other end of the cable, and the correct method of connection is shown in **Figure 4.** Suitable edge connectors are readily available, but these are not usually fitted with polarising keys. It might be possible to add these, or the top and bottom of the connector can be clearly marked as such.

Setting up

Due to offsets in the comparator section of the converter the circuit tends to underread slightly, and it is to correct this that the simple zero adjustment circuit is included. To give VR1 the correct adjustment a small input voltage (about 10 millivolts) is applied to the input of the circuit so that a potential of 5 millivolts is produced at the input (pin 6) of IC1. VR1 is then adjusted so that the returned reading flickers between 0 and 1. If suitable test gear to permit this setting up procedure is not available, good accuracy will still be obtained if VR1 is simply set at almost maximum resistance (fully anticlockwise).

In order to use the converter at high speed it is necessary to resort to machine code programming. However, BASIC can handle something in the region of one hundred conversions per second, and this is adequate for many purposes. To set up PA2 as an output, 4 must be written to the port A data direction register at address 56578 (ie POKE 56578,4). PA2 is then controlled by writing 0 or 4 to address 65678. A value of 0 sets PA2 low and selects input 1; 4 sets PA2 high and selects input 2.

The converter is read at address 56577.



First POKE 0 (or any other legal quantity) to this address to produce a conversion and then PEEK the same address to read the converter. If readings are being taken in rapid succession the POKE is only needed before the first reading is taken, since reading the port, as explained earlier, provides a "start conversion" pulse to the converter.

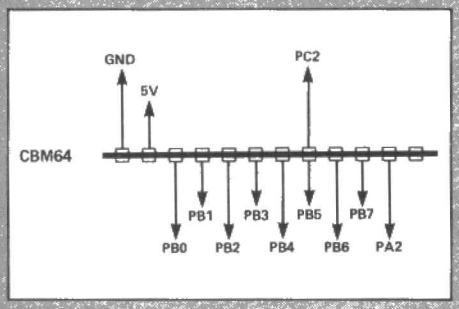


Figure 4a. Connections to CBM 64.

The following short program will therefore continuously read input 2 and print the returned readings down the left hand side of the screen.

```
10  POKE 56578,4
20  POKE 56576,4
30  POKE 56577,0
40  PRINT PEEK(56577)
50  GOTO 40
```

VIC-20

The unit can be used with the VIC-20, and a suitable method of connection to the user port is included in **Figure 4.** In order to set up PA2 as an output a value of 4 is POKEd to address 37139. Then either 0 is POKEd to address 37137 to select input 1, or 4 is used to select input 2. To set CB2 (the

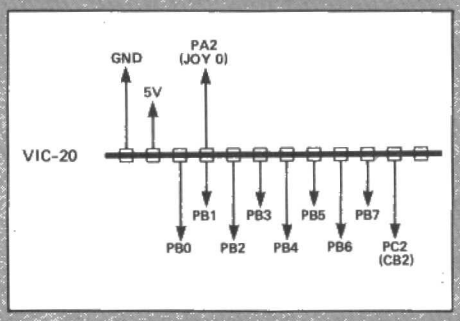


Figure 4b. Connections to VIC 20.

nearest equivalent of PC2 as a pulsed output 160 is written to address 37148. However, a "start conversion" pulse will only be generated after a write operation to port B, and POKE 37136,0 must be used to produce such a pulse prior to reading the converter (which is also at address 37436). The following short program will therefore continuously read channel 1 and print the returned values down the left hand side of the screen.

```
10  POKE 37139,4
20  POKE 37137,0
30  POKE 37148,160
40  POKE 37136,0
50  PRINT PEEK(37136)
60  GOTO 40
```

■

PARTS LIST

Resistors (all ¼ watt 5%)

R1	390R
R2	47k
R3	680k
R4	1k
R5	10k
R6,7,8	8k2 (3 off)

Potentiometer

VR1	1 M0 0.1W horizontal preset

Capacitors

C1	100p ceramic plate 2%
C2	1u 63V axial elect
C3,4	100n polyester (2 off)
C5	10n polyester
C6	100n ceramic

Semiconductors

IC1	ZN449 (or ZN448 or ZN447)
IC2	555
IC3	4051BE
D1,2	1N4148 (2 off)

Miscellaneous

Printed circuit board; 16-pin DIL IC socket and 18-pin DIL IC socket; 10-way ribbon cable; 2 by 12-way 0.156 inch edge connector; Solder, etc.

THE THIRD DI

Mike James puts the principles of three dimensional computer graphics into perspective – with some practical examples you can try out for yourself.

The subject of three-dimensional graphics is the most exciting – and most difficult part of graphics. Most of the impressive examples of computer graphics that appear on television and in magazines are computer representations of three-dimensional objects. Many programmers are attracted to graphics by these stunning examples but soon discover that even the most powerful of today's micros cannot manage anything like the complexity or quality of the published results.

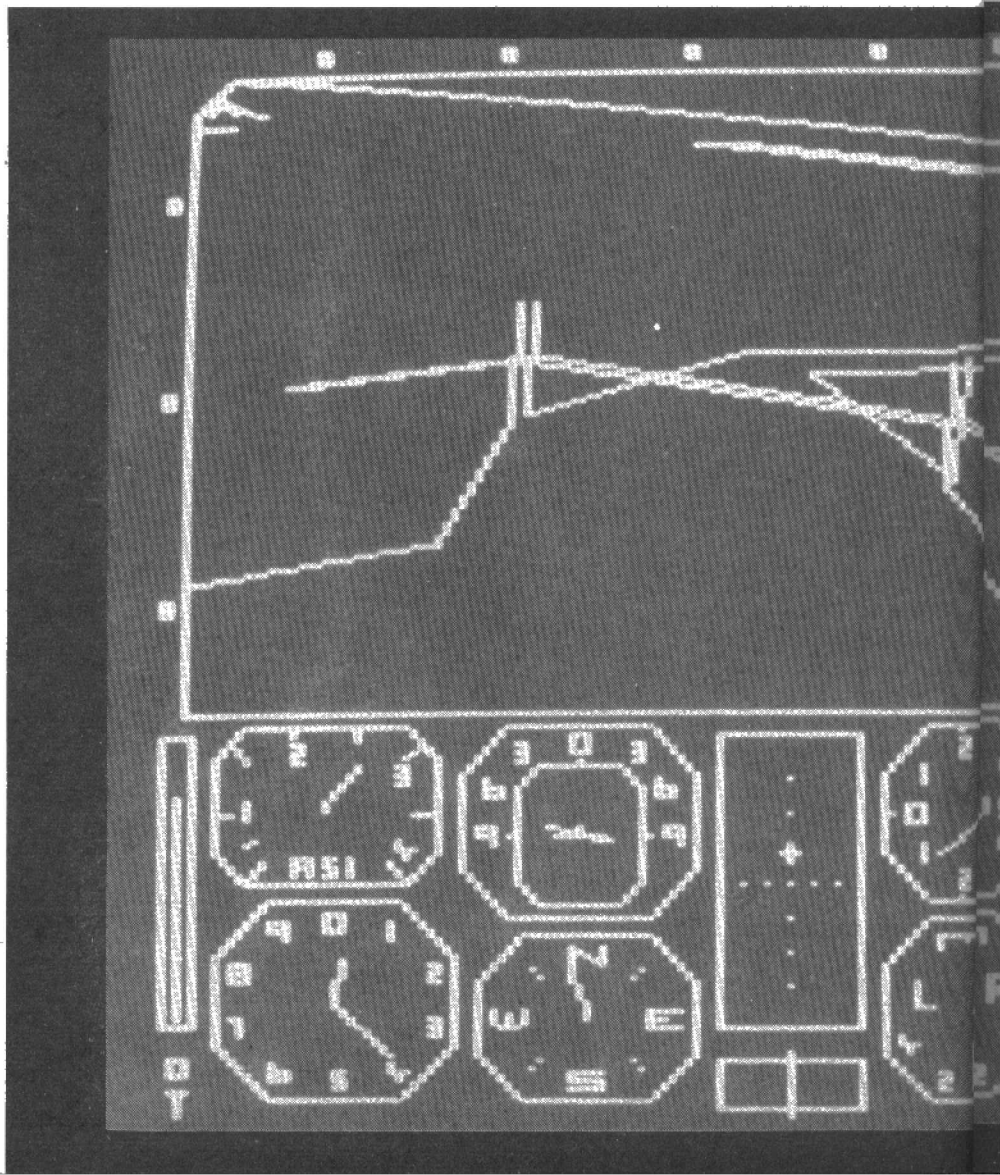
The difficulty in producing images of the highest quality is only partly due to the inadequacies of micro hardware. The biggest culprit is the incredible amount of data needed to define the simplest three-dimensional object. This 'data explosion' makes even trivial operations very time-consuming. It is possible to produce remarkable 'one off' displays using special tricks that rely mainly on identifying and exploiting regularities in the shapes that are being displayed. It is even possible occasionally to find ways of using these regularities to produce moving three-dimensional images but this is the exception rather than the rule.

Despite limitations of hardware, the principles of three-dimensional graphics are not at all difficult and are nothing more than extensions of the ideas encountered in two-dimensional graphics. Rather than look at one or two of the special 'trick' examples of 3D graphics this article describes the general principles. However, the emphasis is not all on the theory and towards the end you will find a few programs that will enable you to experiment with three dimensions. Next month this experience will be used to construct a program that allows a three-dimensional object to be viewed from any position.

Classical and artistic

To a certain extent the ideas described in this article come from the 'classical graphics' used on large computers and expensive high resolution colour displays. In practice, the sort of 3D graphics that turn up in games etc. has more in common with the way an artist or draughtman uses lines on a flat sheet of paper to produce the appearance of depth. True 3D uses a complete description of a shape in terms of the space that it occupies to produce a two-dimensional representation of what would be seen from any particular 'viewpoint'. For example, you might record all of the information concerning the shape of a house and then choose to view it from above, giving a floor plan, or from the front. The important point is that in true three-dimensional graphics the viewpoint can be changed with complete freedom thus allowing the user to examine the object and gain a good impression of what it is like. On the other hand, the 'artistic' method of three-dimensional graphics would use the techniques already introduced in these articles, to produce a floor plan or a front elevation of the house, or whatever, in much the same way that you would draw the same view using pencil and paper. Although such drawings can be made 'freehand' using nothing but intuition this doesn't mean that there are no principles that can be used to make them into effective three-dimensional representations. These guidelines will be the subject of a future Micrographics when the prospect of moving three-dimensional images on a micro are considered.

Using two-dimensional graphics to give the impression of three dimensions is an extremely useful and powerful technique for micrographics but it does have the disadvantage that the user's point of view is

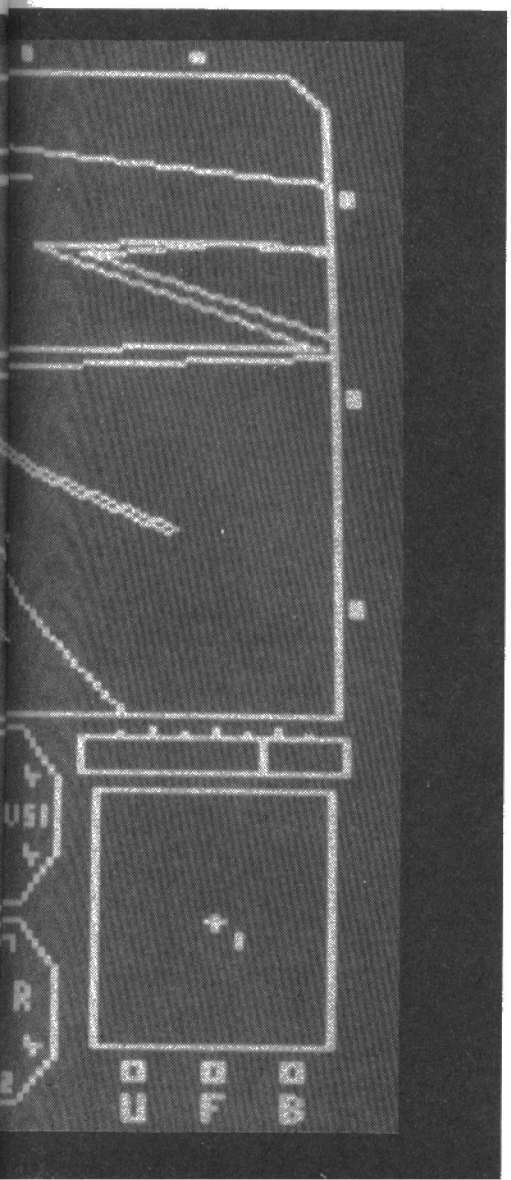


MENSION

more or less fixed. The great advantage of the classical method is that the user's viewpoint can be varied at will and the computer can be used to explore and examine the object as if it was real.

Points, lines and planes

The point and the line play an equally important role in three dimensions as they do in two. In two dimensions two points define the location of a straight line and a number of connected lines define a shape. The same is true in 3D graphics, but each point requires three co-ordinates, usually denoted x,y,z. However the most important feature of 3D graphics is that the two-dimensional shapes defined by a number of connected lines can be used to enclose a volume and hence form a solid shape. For example, a cube can be defined by putting together six square faces, each defined by four lines.



There are two slightly different types of 3D representation, depending on whether or not the faces of a solid object are considered transparent or opaque. If they are transparent, then all the faces of an object are visible at the same time and it appears as a 'wire frame' (see **Figure 1a**). If the faces are opaque then one face will be obscured from others that are further from the viewing position (see **Figure 1b**). Producing 3D images with opaque faces is a very difficult problem involving so called 'hidden line removal' algorithms.

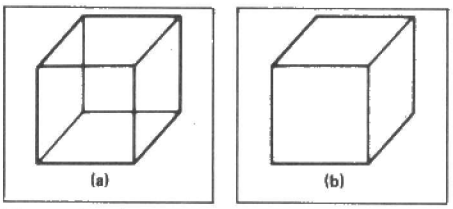


Figure 1a. (left) shows the appearance of a 'transparent' cube and Figure 1b an opaque cube.

Although detecting and removing lines that are hidden from view by other faces seems like a trivial problem it is in fact extremely time-consuming and well outside the practical range and speed of current micros. For the rest of this article the hidden line problem will be ignored and only wire frame displays will be considered, but it is worth saying that it is usualy necessary to construct a wire frame display before converting it to an opaque representation by removing the hidden lines.

The problem of finding a data representation suitable for 3D graphics is more complex than you might expect. Obviously the method of point and line files used in two-dimensions can easily be extended to three by adding an extra co-ordinate to each point in the point file. However, although the point and line file are sufficient to define an object, they do not always present the information needed in the most accessible form. For example, given the co-ordinates of the eight corners of a cube stored in the point file and the 12 edges in the line file, it is a difficult job to work out which lines in the line file go together to form each face of the cube. This difficulty could be alleviated by adding a 'face file' to the data structures. Each entry in the face file would simply be a list of the lines in the line file that make up each face. Once the face file has been added then it is easy to see that it might be worth recording other information about each face at the same time, for example its colour or texture could be used to create a shaded solid representation (given suitable hardware!). Fortunately wire frame representations of objects can be constructed without any additional complications to the point and line.

Now that the method of storing the data has been decided on it is worth giving an example to illustrate the 'data explosion' problem of 3D graphics. The points that have to be stored in the point file to represent the corners of a 'unit cube' can be seen in **Figure 2.** The three numbers written next to each corner are the x co-ordinate, y co-ordinate and the z oc-ordinate respectively. Thus the point file consists of three arrays X, Y and Z and the Ith point is given by x(I), Y(I) and Z(I).

It is obvious that even a simple cube needs 24 (=8 corners x 8 co-ordinates) array elements to represent them. There are also twelve lines that have to be stored in the line file to complete the description of the cube. Each line is defined by a start point and an end point and so the twelve lines take 24 array elements to store making a grand total of 48 arrays elements. This should be compared to the 16 elements needed in two dimensions to store the description of a square. It doesn't take much imagination to see that getting the data for an object with a complex shape into the system is going to be a major problem.

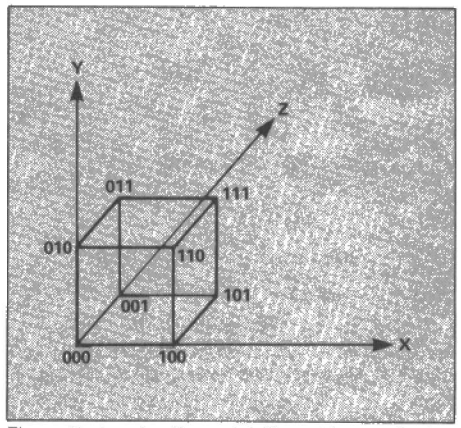


Figure 2 showing the point file entries to display a cube.

3D transformations

The matrix method of rotating, translating and scaling two-dimensional objects is easy to extend to three dimensions. The only real change is that instead of using a column vector with three elements we use one with four and the three-by-three transformation matrices become four-by-four. That is, each point is represented by a column vector of the form:

```
| x |
| y |
| z |
| 1 |
```

where once again the final 1 is included to allow translations to be implemented as matrix operations. As an example of a four-by-four transformation matrix consider:

```
| COS T   0   SIN T   0 |
|   0     1     0     1 |
| -SIN T  0   COS T   0 |
|   0     0     0     1 |
```

which produces a rotation through an angle T about the y axis if applied to all the points in the point file. Similar matrices can be put together for rotation about any axis, translation, and scaling; but these are in general not as useful as in the two-dimensional case and it would take several magazine pages to list each type! What is important is the idea that a 3D object can be rotated, translated and scaled for viewing simply by multiplying by an appropriate transformation matrix rather than the particular form of the matrix used.

Projections

Using 3D transformations it is possible to alter the viewpoint of any object by rotation, translation or scaling but there still remains a fundamental problem – how can a three-dimensional object be drawn on a two-dimensional screen. Put more forcibly, this problem is essentially that each point on the object has three co-ordinates but a point on the display screen has only two co-ordinates. Clearly what is

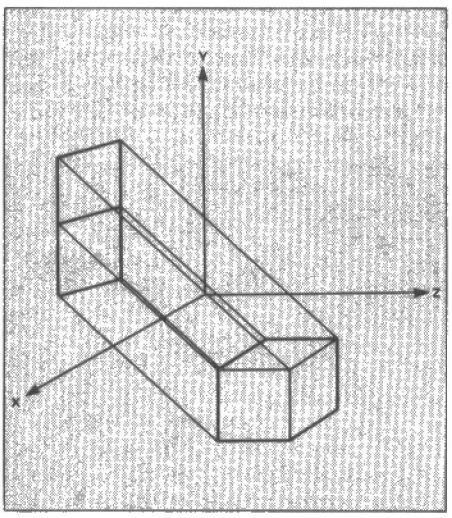


Figure 3. Parallel projection of cube onto X, Y plane.

needed is a way of reducing the three-co-ordinates to two in such a way that the points plotted on the display screen look like the object when viewed from the given position. You might think that this is difficult but in fact it is nothing more than another extension of the matrix transformation methods. If you multiply a four element column vector by a four-by-three matrix the result is a three-element column vector, ie a column vector of the type used to represent a point in two dimensions. Such a transformation matrix is called a 'projection' and viewing three-dimensional objects on a two-dimensional screen is all a matter of finding and using the correct projections. Very often it is inconvenient to use a four-by-three transformation matrix and instead the usual four-by-four matrix is used to present a projection by ensuring that the z co-ordinate of the resulting four element column vector is always zero.

In practice only two types of projection are used with any frequency – the parallel projection and the slightly more sophisticated 'perspective' projection. Both of these projections are described below.

The parallel projection

The parallel projection has been used by draughtsmen for many years to illustrate a three-dimensional shape by way of a number of two-dimensional drawings. If you have ever studied technical drawing then the familiar isometric, trimetric and dimetric projections are all examples of parallel projections. A parallel projection is produced by choosing direction and extending lines parallel to the direction from each corner of the object until they intersect a plane that represents the viewing screen. This is an idea that is easier to understand by way of a diagram! **Figure 3** shows a parallel projection of a cube onto the plane defined by the x and y axis. By altering the position of the plane and the direction of projection, a wide range of views of the object can be obtained. In fact a parallel projection is a rough approximation of what you would see if you positioned your eye on the far side of the viewing screen and looked along the direction used to draw the parallel lines.

The projection matrix that produces a parallel projection on the plane defined by the x,y axis in a direction given by XP,YP,ZP is

```
| 1  0  -XP/ZP  0 |
| 0  1  -YP/ZP  0 |
| 0  0     0    0 |
| 0  0     0    1 |
```

which, as already mentioned, is written for simplicity in a form that produces a three-dimensional result but with the z co-ordinate always 0. The only complication is the way that XP,YP,ZP determine the direction of the projection. If you imagine a line drawn from the origin (ie 0,0,0) to the point XP,YP,ZP, then this line, taken from the origin out to the point, is the direction to which all of the projection lines are parallel.

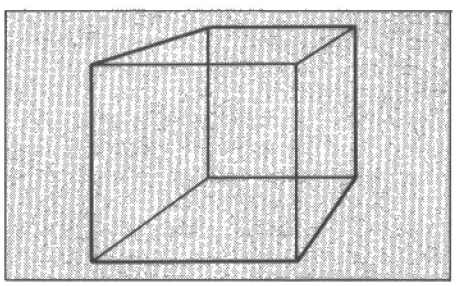


Figure 4. Illustrating the effects of perspective.

Experimenting with parallel projections

In preparation for the complete three-dimensional viewing program in next month's article, the following short program uses the projection matrix given in the previous section to display a parallel projection of a cube on the plane formed by the x,y axis. Although written in BBC BASIC none of the BBC Micro's special features are used and it should be easy to convert it to other micros. The subroutine structure can be seen in **Table 1.**

Table 1

line number	use
10-70	main program
1000	set up line and point file for object and general initialisation
2000	get the display parameters – in this case the direction of projection
3000	apply projection matrix to every point in the point file
4000	plot the result using the information in the original line file and the new 2D point file

The subroutines used in this program will be incorporated into the full three-dimensional view routine to be given next month so it is worth examining each one in detail. Subroutine 1000 uses a list of DATA statements to store the necessary information to define a unit cube in the point file, consisting of X(I),Y(I),Z(I) and the line file, consisting of S(I) and E(I). Each data statement between 1010 and 1080 contains the co-ordinates of a single corner of the cube. Similarly each data statement between 1100 and 1210 contains the index of the start and end points of an edge of the cube.

To change the graphics data to represent some other more interesting object then you also need to know that P, in line 1300, is set to the number of points in the point file and L, in line 1310, is set to the number of lines in the line file. Lines 1320 to 1370 read the data from the DATA statements and into the arrays. The only complication here is that having a unit cube positioned at the origin may not give a reasonable sized image when projected on the plane of the x,y axis and so all the co-ordinates are multiplied by 100, to scale the cube to 100 by 100 by 100, and 500 is added to translate the corner that was at the origin to the point 500,500,500. If you are using this program with another micro then you will find it necessary to alter these values – try multiplying by roughly 1/10th of the largest screen co-ordinate, adding about half of the maximum screen co-ordinate and then adjust the value by trial and error.

The only other job performed by subroutine 100 is to define the arrays A(I) and B(I). These are used to hold the x and y co-ordinates of the results of the projection transformation so that the original data in the point file is left unchanged. Subroutine 2000 simply reads in the x,y and z co-ordinates of the point that defines the direction of projection into XP,YP and ZP respectively. Subroutine 3000 applies the actual projection transformation. If you work out the matrix multiplication involved it boils down to:

```
a=x-z*xp/zp
and
b=y-z*yp/zp
```

where a and b are the co-ordinates of the point on the screen corresponding to the point on the object at z,y,z and the direction of projection is given by xp,yp,zp. The final subroutine simply plots the lines in the line file but using the results of the projection stored in A(I) and B(I) as the point file rather than X(I),Y(I),Z(I).

Now for another problem: finding values of XP,YP and ZP that give good looking results. It is a very human inability to imagine the effect of any given change in the direction of projection when specified by co-ordinates in this way. It may help to know that entering a value of 1,1,1 gives a direction of projection that is parallel to a diagonal of the cube. If you start from the value .7,.7,1 then you will see a type of projection known by draughtsmen as a 'cavalier' projection. Notice that only the ratios of XP, YP and YZ are important in determining the direction of projection. For example, 7,7,10 would produce the same cavalier projection as .7,.7,1. Also notice that by changing the angle of projection you cannot move closer or rotate the cube to examine any face; all you can do is to alter the amount of the 'front' facing sides that are visible.

```
  10 REM Projection 1
  20 MODE 4
  30 GOSUB 1000
  40 GOSUB 2000
  50 GOSUB 3000
  60 GOSUB 4000
  70 GOTO 40

1000 DIM X(20),Y(20),Z(20),S(20),E(20)
1010 DATA 0,0,0
1020 DATA 0,0,1
1030 DATA 0,1,0
1040 DATA 0,1,1
1050 DATA 1,0,0
1060 DATA 1,0,1
1070 DATA 1,1,0
1080 DATA 1,1,1
1100 DATA 0,4
1110 DATA 4,6
1120 DATA 6,2
1130 DATA 2,0
1140 DATA 1,5
1150 DATA 5,7
1160 DATA 7,3
1170 DATA 3,1
1180 DATA 0,1
1190 DATA 4,5
1200 DATA 6,7
1210 DATA 2,3
1300 P=8
1310 L=12
1320 FOR I=0 TO P-1
1330 READ X(I),Y(I),Z(I)
1340 NEXT I
1350 FOR I=0 TO L-1
1360 READ S(I),E(I)
1365 X(I)=100*X(I)+500
1366 Y(I)=100*Y(I)+500
1367 Z(I)=100*Z(I)+500
1370 NEXT I
1500 DIM A(20),B(20)
1510 RETURN

2000 PRINT TAB(0,1);
2010 INPUT "XP,YP,ZP",XP,YP,ZP
2020 RETURN

3000 FOR I=0 TO P-1
3010 A(I)=X(I)-Z(I)*XP/ZP
3020 B(I)=Y(I)-Z(I)*YP/ZP
3030 NEXT I
3040 RETURN

4000 CLS
4010 FOR I=0 TO L-1
4020 MOVE A(S(I)),B(S(I))
4030 DRAW A(E(I)),B(E(I))
4040 NEXT I
4050 RETURN
```

Program to display projection of a cube on X, Y axis.

Perspective

Although parallel projection produces fairly good impressions of 3D objects, it suffers from one important deviation from reality: in real life the apparent size of something depends on how far away it is from the observer. For example, if you were to view a wire cube from a few feet then the top front horizontal edge would appear slightly larger than the top rear horizontal edge – see **Figure 4.** As you move further away from the cube then the difference between the two edges becomes less, and as you move closer it becomes exaggerated. This is of course the phenomenon of 'perspective' and it is something that we all take for granted both in our everyday viewing of the world and in traditional pictures and drawings that attempt to produce a sense of realism.

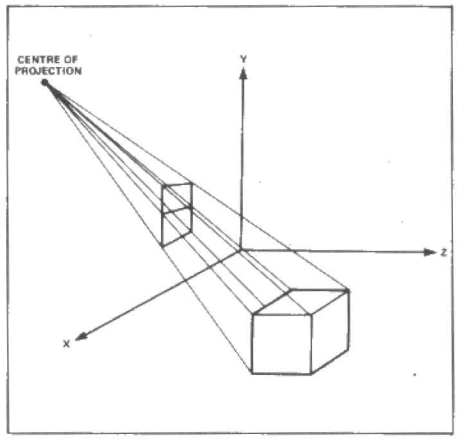


Figure 5. Cube projected onto X, Y axis plane.

It is not difficult to alter the method of parallel projection to produce a perspective projection. The only real difference is that instead of the lines used in the projection running parallel, they converge to a special point – called the 'centre of projection'. Roughly speaking, the centre of projection is the position that an observer would occupy to see the same results produced by the perspective projection. Once again the actual display on the two-dimensional screen is arrived at by determining where the lines of projection meet a given plane. All this can be seen in **Figure 5,** where a perspective projection of a cube onto the plane formed by the x,y axis is shown.

If you think about it for a moment, you will realise that the co-ordinates derived from the intersection of the lines of projection are going to be used to plot lines on a TV screen which will then be viewed by an observer. To give natural perspective the ratio of the distance that the viewer is from the TV screen to its size should be the same as the ratio of the centre of project to the size of the plane of projection. For example, if the viewer is normally 24" from a 12" TV screen and a 12" line corresponds to one 400 units long in terms of the graphics co-ordinate system, the centre of projection should always be 800 units from the plane of projection. If it is much closer or much further away then the perspective will seem unnatural.

The perspective projection matrix is:

–ZC	0	XC	0	
0	–ZC	YC	0	
0	0	0	0	
0	0	1	–ZC	

where XC,YC,ZC is the position of the centre of projection. This transformation is a little different to the others that we have used in that it alters the value of the final 'dummy' fourth co-ordinate. If you look back to the Micrographics article that first introduced transformations you will see that the four co-ordinates used to define position of either a three- or a two-dimensional point, the final co-ordinate MUST be 1. This can be ensured by dividing all of the co-ordinates by the value of the final co-ordinate. That is, if the transformation gives the result x,y,x,w then the two dimensional point to be plotted is x/w,y/w – thus a perspective projection involves division.

Experimenting with perspective

Changing the parallel projection program given in the last section only involves subroutines 2000 and 3000. If you apply the perspective projection to the column vector x,y,z,1 then you will find that the co-ordinates of the corresponding two-dimensional point is:

$$a = \frac{x*zc + z*xc}{z - zc}$$

and

$$b = \frac{-y*zc + z*yc}{z - zc}$$

Changing subroutine 3000 to use these two equations is easy and subroutine 2000 now reads in the co-ordinates of the centre of projection:

```
2000  PRINT TAB(0,1);
2010  INPUT "XC,YC,ZC",XC,YC,ZC
2020  RETURN
3000  FOR I=0 TO P-1
3010  A(I)=-X(I)*ZC+Z(I)*XC
3020B   (I)=-Y(I)*ZC+Z(I)*YC
3025  W=Z(I)-ZC
3026  A(I)=A(I)/W
3027  B(I)=B(I)/W
3030  NEXT I
3040  RETURN
```

Here the difficulty is in finding the position of the centre of projection that gives the desired projection. However, if you start from 400,550,1000 and move the point around in steps of around 50 units then you will be able to see the range of possible perspective views – including some that are extremely exaggerated.

. . . and next month

The fundamental tool of three-dimensional graphics is the projection but, although the two example programs given are suitable for experimentation, they are not really very easy to use. Next month's Micrographics will give a complete three-dimensional viewing program that can be used to explore and examine any object of your choice.

68705 EPROM blower

Part 2 of John Williams' universal EPROM programmer. Construction details, software, and hardware are explained

To operate the programmer a blank or partially programmed EPROM is inserted into the socket and the unit connected to a computer or RS232 terminal. The 5 volt supply to the unit is then switched on; the higher programming voltage can be left off until actual programming is required to reduce the risk of accidental programming. The mode of operation is selected by sending a single character. **Figure 6** lists these modes with their respective characters and syntax; either upper or lower case letters being acceptable. The machine will only recognise these letters and the hexadecimal digits 0 to 9 and A to F, in upper or lower case. Any other character, printable or not, including space and carriage return, is treated as a delimiter indicating the end of a character sequence. Carriage return additionally has a special significance in the LIST mode as detailed below. All communication other than the mode control characters of Figure 6 will be in hexadecimal and the above definition of delimiters will ignore any prefixes or suffixes (ie &, $ or H) sometimes used to indicate a hexadecimal number, and will pick out only the part containing the actual number. Hexadecimal numbers must consist of either 2 digits for 8 bit data or 4 digits for a 16 bit address. Only the lowest 11 to 15 bits of the address are actually used depending on the size of the EPROM.

T d :- Test for blank EPROM
L (D hhhh) (D hhhh) C/R :- List
P (d hhhh) d hh d (hh d) :- Program
V (d hhhh) d hh d (hh d) :- Verify
X d :- Exit from programming or verifying

Note:
C/R = carriage return
d = any delimiter
D = any delimiter except C/R
h = one hexadecimal character
Terms in brackets are optional

Figure 6. Control characters and their syntax.

The routine of running the programmer is detailed below; the sections can be done in any order.

To test for a blank EPROM send T followed by a delimiter. After about a minute the green LED will illuminate if the EPROM is blank. If a programmed location is encountered the red LED will illuminate immediately.

To list the contents send L with delimiter optionally followed by one or two 4 digit hex addresses. The first address, if present, defines the beginning of the listing. If no second address is sent then only the contents of the first address is listed. If a second address is sent the listing will continue up to and including this address. If no addresses are sent at all the unit will list all locations from 0000 to FFFF. The listing will not start after the character L or the first address unless a carriage return is sent. This is the only situation where carriage return is treated differently from other delimiters. The format of the listing will be as in **Listing 1** except there only being 8 characters per line. Since the bits of the address above those required by the particular size of EPROM in use will be ignored they can be set to any value for listing purposes. This can be used to provide a printout of the contents at the correct base page address where the EPROM will be in the final application.

To program the EPROM send P followed by a delimiter. An optional 4 digit hexadecimal address with delimiter can then be sent, if it is omitted the programmer will assume the code is to start at 0000. Next send the 2 digit hexadecimal code which is to go in that address. The unit will then program the EPROM and also increment the address by one. If consecutive addresses are to be programmed the data can be entered without re-entering the new address. At any time a new 4 digit address can be entered followed by its code. When the last of the code has been entered the letter X should be sent in order to turn off the programming supply. Whenever this voltage is present the warning LED D4 will be illuminated. The output of the LIST routine meets the requirements of programming and allows duplication or modification of an existing EPROM. The EPROM is listed and the data stored as a string in the computer. This can be modified as necessary and fed back to a blank EPROM preceeded by the character P and delimiter and followed by X.

To verify the EPROM the code to be verified is sent using the same format as for programming but it is preceeded by V instead of P. When the first code is received the green LED will illuminate if it corresponds to the code in the EPROM, if any error is detected the red LED will illuminate and remain so until reset by the letter X or by selecting a new operating mode.

Construction

A design for a printed circuit board is available, see page 49. For economy it is single sided and this inevitably results in the need for links to be fitted in the places shown. The author's prototype was constructed following a very similar layout but using plain (not tracked) Veroboard; this form of construction although perhaps more time consuming is equally suitable. Either way construction is quite straightforward, a 28 pin zero insertion force socket should be used for the EPROM and normal sockets for the other ICs. The EPROM selector is constructed from a plug and

```
100   REM  DEMONSTRATION PROGRAMME
110   REM ----Test for blank----
120   PRINT AT RS232 "T "
130   PRINT  When red LED lights press return
140   INPUT A$
150   REM ----Verify----
160   PRINT AT RS232 "V 0080 A6 20 20 22 0090 B6 11 "
170   PRINT  Check green LED lights; press return
180   INPUT A$
190   REM ----List----
200   PRINT AT RS232 "L 0080 07FF "
210   INPUT AT RS232 P$
220   PRINT P$
230   GOTO 210
500   REM ----Programme----
510   PRINT AT RS232 "P 0010 12 34 0020 EF X "
```

Figure 7. When used with the EPROM containing the 68705P3 code this program illustrates the operation of the programmer.

Listing 1.

```
0080  A6 20 20 22 A6 0D AD 1E A6 0A 20 1A B6 10 AD 02
0090  B6 11 B7 18 46 46 46 46 AD 02 B6 18 A4 0F A1 0A
00A0  25 02 AB 07 AB 30 B7 22 00 02 FD 13 02 AE 08 AD
00B0  1B 36 22 24 04 12 02 20 04 13 02 9D 9D 9D 9D 9D
00C0  9D 9D 9D 5A 26 E9 AD 04 12 02 AD 00 A6 EA 4A 9D
00D0  9D 9D 26 FA 81 B6 10 B0 12 26 04 B6 11 B0 13 B7
00E0  25 A6 01 BB 11 B7 11 A6 00 B9 10 B7 10 B6 25 81
00F0  3F 20 3F 21 15 02 81 B6 10 AD 02 B6 11 AE 80 1F
0100  01 49 24 02 1E 01 16 02 17 02 56 24 F2 81 FF FF
0110  B6 20 27 FC A1 01 26 01 83 B6 31 48 48 48 48 BB
0120  32 BC F0 1F 29 B6 20 27 FC A1 01 26 01 83 A1 04
0130  27 03 1E 29 81 B6 31 48 48 48 48 BB 32 B7 14 B6
0140  33 48 48 48 48 BB 34 B7 15 BC F0 FF FF FF FF FF
0150  3F 20 3F 21 A6 73 B7 01 A6 FF B7 05 15 01 A6 02
0160  B7 02 A6 FF B7 05 A6 0E B7 06 A6 45 B7 09 3F 04
0170  3F 23 3F 24 3F 25 3F 29 B6 01 A4 73 AA 70 B7 01
0180  9C 9A B6 20 A1 01 26 FA 10 01 12 01 BD F0 B6 31
0190  A1 54 26 03 CC 01 B0 A1 4C 26 03 CC 01 F0 A1 50
01A0  26 03 CC 02 50 A1 56 26 03 CC 02 90 20 D2 FF FF
01B0  BD F0 3F 10 3F 11 A6 FF B7 12 B7 13 A6 13 B7 01
01C0  BD F7 B6 00 A1 FF 26 07 BD D5 26 F4 11 01 83 13
01D0  01 83 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
01F0  A6 00 B7 10 B7 11 A6 FF B7 12 B7 13 00 24 23 CD
0200  01 23 0E 29 1D B6 14 B7 10 B7 12 B6 15 B7 11 B7
0210  13 00 24 0E CD 01 23 0E 29 08 B6 14 B7 12 B6 15
0220  B7 13 A6 13 B7 01 A6 08 B7 2A BD 8C BD 80 BD 80
0230  BD F7 B6 00 BD 92 B6 20 26 06 BD D5 26 03 BD 84
0240  83 3A 2A 26 E9 BD 84 20 DD FF FF FF FF FF FF FF
0250  A6 FF B7 04 A6 77 B7 01 A6 00 B7 10 B7 11 CD 01
0260  23 0E 29 08 B6 14 B7 10 B6 15 B7 11 BD F7 CD 01
0270  10 B7 00 A6 4F B7 01 A6 FF AE 17 5A 26 FD 4A 26
0280  F8 A6 77 B7 01 BD D5 20 D5 FF FF FF FF FF FF FF
0290  A6 13 B7 01 11 01 A6 00 B7 10 B7 11 CD 01 23 0E
02A0  29 08 B6 14 B7 10 B6 15 B7 11 BD F7 BD D5 CD 01
02B0  10 B1 00 27 E7 10 01 13 01 20 E1 FF FF FF FF FF
02C0  2F 03 03 29 01 80 A6 08 B7 23 3F 26 A6 9A B7 08
02D0  1F 09 1D 09 12 29 80 FF FF FF FF FF FF FF FF FF
02E0  4F 2E 01 43 46 36 26 1F 09 3A 23 27 05 A6 66 B7
02F0  08 80 1C 09 13 29 BE 21 B6 26 A1 30 25 1F A1 3A
0300  24 04 A0 30 20 0C A4 DF A1 41 25 11 A1 47 24 0D
0310  A0 37 E7 31 A3 05 27 02 3C 21 11 24 80 A1 0D 27
0320  16 A1 50 27 1D A1 4C 27 19 A1 56 27 15 A1 54 27
0330  11 A1 58 27 0D 20 02 10 24 B6 21 B7 20 27 02 14
0340  02 80 B7 31 A6 01 B7 21 80 FF FF FF FF FF FF FF

0784  00                    07F8 02 E0 02 C0 01 5C 01 50
```

socket system normally sold for inter PCB connection. Four 10-way straight plug strips are cut down to six pins each and are soldered directly to the board. A 6-way free socket is constructed by cutting down a 10-way shell. The sockets to use in the shells are supplied separately and although designed for crimping can with care be squeezed with pliers and soldered to multistrand wires. These are soldered to the PCB as shown on the layout, they are loomed and tied to the board to ensure there is no chance of the socket being accidentally mated the wrong way round.

The five RS232 wires are soldered to the numbered positions on the board and to the corresponding pins of a 25-way free plug. The pin numbers shown are for a standard 25-way D type plug to mate to a computer. If the unit is used with a terminal instead of computer then the wires should be taken to a 25-way socket using the numbers shown in brackets on the circuit diagram. This transposition is necessary since the send pin number on a computer is the receive pin number of a terminal and vice versa.

Software

The machine code required to run the 68705P3 is shown above. The area below 0080 is not used and the section between 0350 and 07F7, including the Mask Option Register at 0784, contains 00. This code must first be put into a 2716 for which the reader requires access to an existing EPROM programmer. The code is then copied into the internal EPROM of the 68705P3 using the circuit and procedure given in the October 1983 article. Once this is done the 68705P3 is plugged into the board and the programmer is ready for use.

As many readers will no doubt wish to understand the operation of the program, and perhaps modify it to their particular requirements, the code is shown in mnemonic form in **Figure 9.**

The original code was not written on an assembler but was hand assembled with the aid of an interactive disassembler program on a 6800 microprocessor system. It should be straightforward to put onto the assembler given in the January 1984 *E&CM* provided labels are inserted instead of addressed for jumps and branches. Standard mnemonics are used except for BRSET, BRCLR, BSET and BCLR which are abbreviated to BRS, BRC, BSE and BCL respectively. To aid development the main sections of the program were arranged to start at convenient hex boundaries and the resulting inter-program gaps left blank containing FF. These can be left out in an assembled version but the NOPs where used must remain as they form parts of timing loops.

A series of subroutines are employed to perform the majority of the basic functions. These are called as required from the main part of the program. A limited number of RAM locations are used for specific purposes as listed in **Figure 10.** The upper part of the RAM is used as the machine stack. Some of the more interesting features of the program are described below.

The program starts at 0150 being the vectored restart address. After initialising the ports and setting the EPROM to an inactive state, the program loops around at 0182 until memory location 20 indicates that a character and delimiter have been received. The program will then jump to the selected routine where the EPROM will be set to the required state and the appropriate action of reading or programming undertaken.

```
0010 / 11 Start address
0012 / 13 Stop address
0014 / 15 Current address
0018 Temp
0020 No. of characters before delimiter
0021 No. of characters currently received
0022 Character being printed
0024 Bit7 = C/R flag
0025 Temp
0026 Received Byte
0029 Flags:-
     Bit1 = byte being received
     Bit7 = No address present
002A List: column count
0031 to 0034 Input buffer
```

Figure 8. RAM allocations.

The input of data from the RS232 is handled by means of interrupts and timer requests. The leading edge of a start bit triggers an interrupt request which then starts the timer off for a duration of 1.5 bit intervals. On each timer interrupt the data line is read and stored, the timer is restarted for a duration of 1 bit interval. When eight bits have been read the byte is complete and a flag is cleared to allow the whole process to be repeated. The received byte is analysed and stored in the correct location in the input buffer. Memory location 21 counts the number of characters received, when a delimiter is received this number is transferred into memory 20. Being interrupt driven the reading process

can continue while other tasks, such as programming of the EPROM, are going on with only a negligible loss of time.

The output of data to the RS232 does not use interrupts but instead has a software delay loop to define the period of each bit sent. Although designed for a 300 Baud rate the code can be adapted for 1200 Baud by altering the transmission delay duration – held in memory location 00CD – from EA to 36. The receive rate can be altered by changing the prescaler divide ratio held in memory location 016B from 45 to 43.

Users may also wish to alter the number of codes the List routine prints on a line. This is currently set at 8 so as to be convenient for the majority of home computers' screens with from 32 to 40 characters per line. Memory location 0227 can be changed from 08 to 10 to give 16 codes per line being a better format for printing. The List routine ends lines with carriage return and line feed. The line feed can be removed if required by changing memory location 0086 from AD to 20.

Testing

The wiring once completed should be carefully checked. The programmed 68705P3 and the other four ICs can then be inserted and the 5 volt supply switched on. The current consumption is about 100mA without the EPROM fitted; the supply should be capable of delivering at least 200mA to allow for the EPROM. The constructor should thoroughly acquaint himself with the operation using Test, List and Verify. If the EPROM used to encode the 68705P3 is available this can be used. **Figure 8** shows a BASIC routine to exercise the unit; it may need adapting to the particular dialect of BASIC in use. Only when the unit is shown to be fully working should the programming voltage be applied and programming be attempted. This can be done at first without the EPROM plugged in; if an oscilloscope is available this can be used to check that the correct programming pulse is present as defined in **Figure 3** (see April *E&CM*).

For maximum safety the programming voltage should be applied after the 5 volts and be removed before the 5 volts. The EPROM should not be inserted or removed until all supplies are off.

PARTS LIST

Resistors

R1	10K
R2	470
R3	470
R4	10K
R5	2.2K
R6	1K
R7	10K
R8	22

Capacitors

C1	27 pF
C2	1 μF
C3	0.1 μF
C4	0.1 μF
C5	0.47 μF
C6	0.1 μF

Semiconductors

IC1	MC1489
IC2	74LS00
IC3	MC68705P3
IC4	74LS164
IC5	74LS164

Diodes

D1	1N916
D2	Green LED
D3	Red LED
D4	Red LED
D5	1N4001

Transistors

TR1	2N2222A
TR2	2N2905

Miscellaneous

SKT1 28-way ZIF; SKT2 6-way inter pcb connector; PL1 to PL4 6-way inter pcb connector.

Modem matters

With the recent price reductions, modems are becoming a popular computer add-on. Liz Gregory takes a look at their operation and surveys a variety of available models.

The film War Games probably did more to make the general public aware of modems than any number of articles in magazines such as *E&CM*. A modem is the essential piece of equipment that allows two remote micros to communicate with each other via a standard British Telecom telephone line. The modem (modulator/demodulator) is responsible for translating the binary data output from a micro into audio tones that fall within the band of frequencies capable of being carried over the Telecom network and for the corresponding recovery of data at the remote end.

As more and more services become available to the micro user, modems are becoming increasingly important in the communications field. Selecting a modem for a micro is, however, not simply a case of finding the cheapest, most reliable unit on the market.

Modem types

Modems fall into two general categories, acoustic coupler and hard wired modems, the latter is also known as direct connect. Acoustic coupler modems have two sockets for the telephone handset to slot into, data being received and transmitted via the 'phone's mouth and ear pieces. Although these sockets are protected by a rubber surround they are still subject to interference from external room noise. Direct connect modems are wired into the telephone line and do not suffer this problem. It is also important to remember the type of 'phone that you have when buying an acoustic coupler and to be especially wary of non-British models as American telephones in particular are slightly different in size. An ill-fitting handset will certainly not improve the quality of data exchange.

Prices of direct connect modems have decreased and their portability has improved with the introduction of models like Dacom's Buzzbox which is light and fairly inexpensive at £79.95. Hard wired modems are, though, subject to approval by the recently formed BABT (the British Approval Board for Telecommunications) as they have to be wired into the British Telecom network.

As for compatability, most users of home micros will find plenty of modems which can be interfaced with their machines provided they feature RS232 interfaces (or equivalent). With regard to data transmission, although most computers convert the user's instructions into the common binary code ASCII, there are variations of this in each machine's make up which require correction. Although these problems are not insurmountable, they do require relevant software or basic programming to overcome them.

The speed at which modems transmit and receive data is specified in a unit's baud rate, which is approximately the number of bits transmitted per second. Information is sent at a certain speed, say 1200 baud and the receiver similarly replies at a predefined rate, say 75 baud. Anyone wishing to use the services of Prestel will need a modem which operates or is capable of being adjusted to reception of transmission at 1200/75 baud.

Computers may simultaneously send and receive information via a single telephone link. This is known as full duplexing, required perhaps when users are communicating with viewdata services. To accomplish this the modem must be of a type capable of supporting both originate

and answer modes. The difference between the modes is in the frequencies used by the modem to represent binary 0 and 1. If both units used the same frequency to transmit data, simultaneous transmission would not be possible as the signals would interface with each other. By allocating two sets of frequencies and preceeding the receive section of the modem with a band pass filer, two way communication becomes possible. This is illustrated in **Figure 1.**

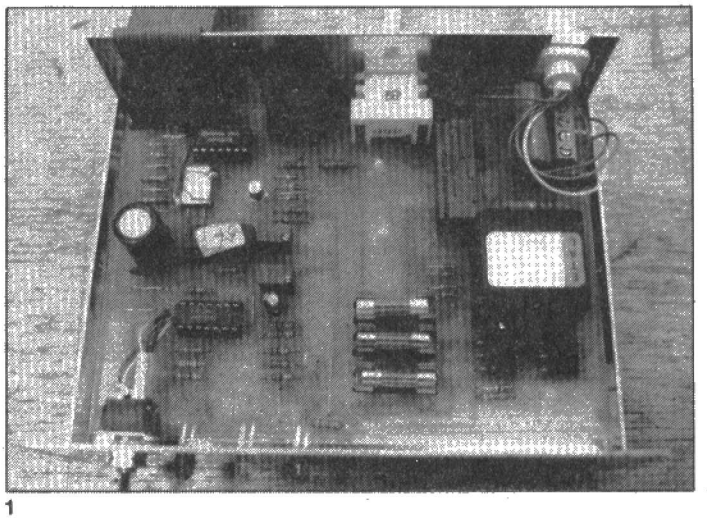
Alternatively when two micros are in communication, data may only be sent in one direction at a time. This form of transmission is known as half duplexing. Many modems have facilities for both types and may be adjusted accordingly.

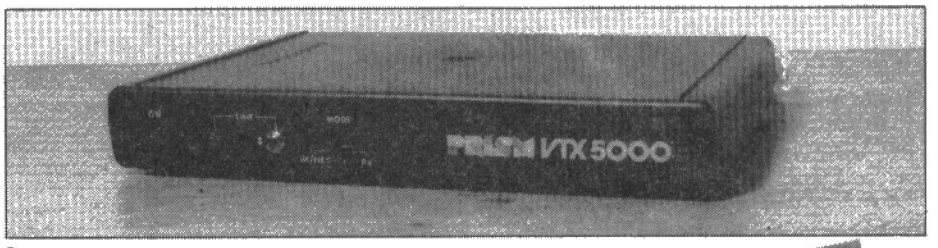
It has become more common now for modems to have built-in auto dial and auto-answer facilities whereby the remote computer will answer a call automatically. Many modems including Watford's new device, Tandata's TM100 and both Display models 2B and 20 modems have this facility built-in. Some companies provide this as a separate package, for example Minor Miracles offer an auto-dial/auto-answer plug-in board for £39.95 excluding VAT.

Naturally, a modem cannot operate without the relevant software. This is normally readily available when the device is initially bought, provision being made for individual machines by the manufacturer. Companies such as Tandata have produced Micropacks where, for example, software for the BBC machine is available on cassette complete with an RS232 connecting cable and a manual. Similarly, Prism provide Comms Packs to be sold with their modems 1000 and 2000 and these include software in ROM form as well as on cassette.

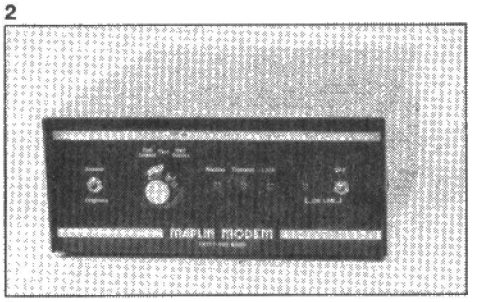
There seems to be a growing trend whereby an overall price package includes the cost of both modem and software. Pace are soon to release their Grapevine modem which will feature a full duplex device 300/300 and 'intelligent' software called Commstar suitable for the BBC. This will use the BBC's memory as a buffer and allows connection to bulletin boards and other BBC's. Some modems like Prism's VTX500 designed specifically for the Spectrum 16/48K come complete with on-board software included in the ROM, a device aimed particularly at the Prestel user. ▶

1. *Inside Watford's new direct connect modem '84.*
2. *The Prism VTX 5000 designed for the Spectrum.*
3. *The Maplin Modem comes as a kit.*
4. *Some modems like the Minor Miracles WS2000 feature facilities for US and European standards.*
5. *Using a modem with an electronic ordering service.*

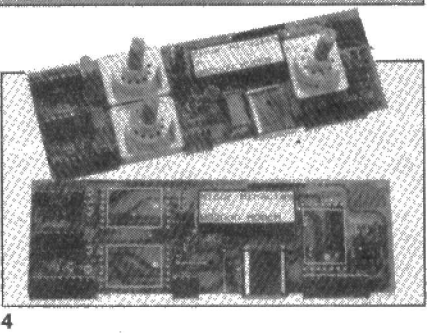

1

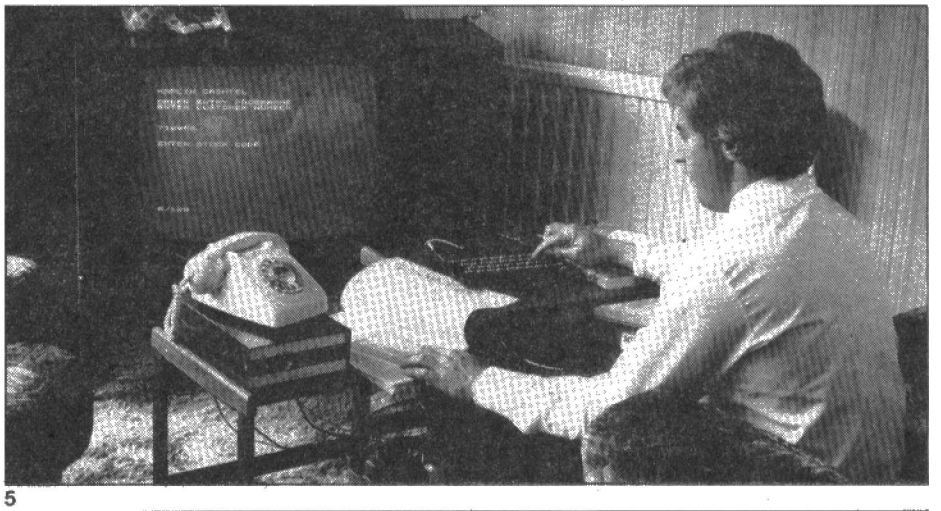


2



3


4


5

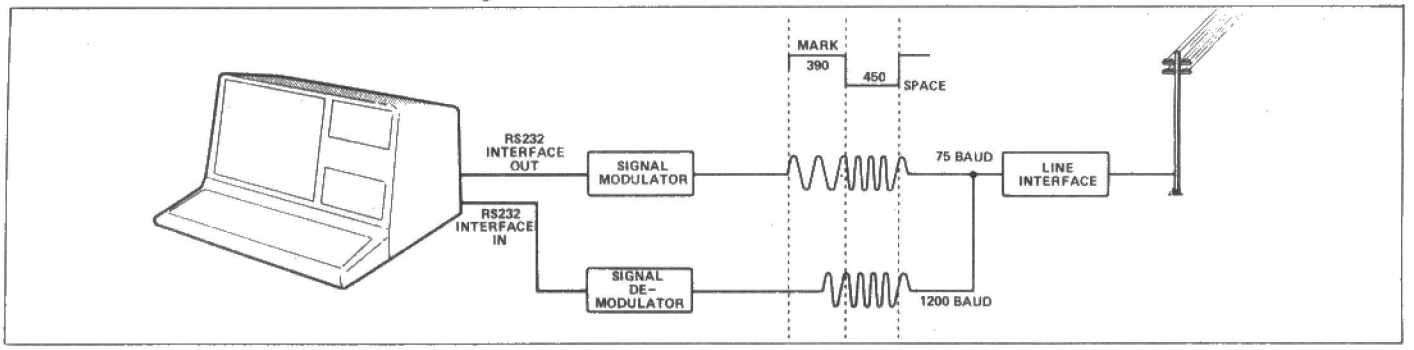


Figure 1. Simultaneous transmission may be achieved by using two sets of frequencies.

Services available

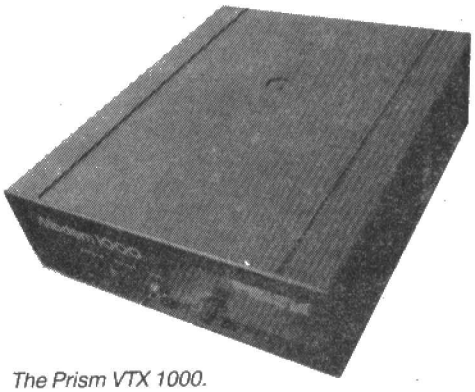
There are a growing number of services available for the micro/modem user. Prestel, which may be accessed by 1200/75 baud modems offers several features including Micronet, which opens up around 30,000 pages of news and views with opportunities for free use of a basic 50 software programs and a 15-20% saving on other software packages. Other services include grocery and banking services like Homelink, Club 403 and even electronic mail services like British Telecom Gold.

Some users may wish to access bulletin boards, which operate at 300/300 baud. These are really in effect computers which provide information, answer questions and general contact between other users. Some companies like Display and Maplin even run an ordering service via their own modems.

One point to bear in mind when choosing a modem is the area over which one wishes to contact other users. Most modems operate under the CCITT standard (the Consultative Committee for International Telephones and Telegraphy) which has a series of standards under which all British and most European modems function. However, as America operates under a slightly different set of standards, if you wish to contact the States it is probably better to obtain a modem that guarantees US protocol. For example, the WS200 modem by Minor Miracles offers this facility as well as a number of different baud rates, which include 1200/75, 300/300 and even 75/1200.

Future Prospects

Modems are undoubtedly getting cheaper and are offering more facilities for your money. It is now possible to get an auto answer, full and half duplexing device complete with software for under £100 and even this price is falling. Watford Electronics are offering their new modem, ready-built and Prestel and British Telecom Gold compatible at a price of £79, an overall figure which includes relevant software. The signs are that direct connect modems will become more portable with some sporting on-board software. Micros could soon start having on-board modems as well and will certainly become cheaper to use as efforts are made to save the user expensive on-line telephone charges. As consumer services grow more popular, there promises to be an improvement in software available and this will be included as part of the overall price package of the modem. ■



The Prism VTX 1000.

Useful addresses

DaCom Systems
16 Alston Drive, Bradwell Abbey, Milton Keynes MK13 9HA.

Display Electronics
32 Biggin Way, Upper Norwood, London SE19 3XF.

Maplin
159-161 King Street, Hammersmith, London W6.

Minor Miracles
PO Box 48, Ipswich IP4 2AB.

Pace
92 Newcross Street, West Bowling, Bradford BB5 8BS.

Prism
30-31 Islington Green, London N1 8BJ.

Tandata
Albert Road North, Malvern, Worcs WR14 2TL.

Watford Electronics
35/37 Cardiff Road, Watford, Herts.

Commercial Services

Cashtel 0702- 552941
Club 403 021-236-3366
Distel 01-679-1888
Homelink (Nottingham) 0602-419-393
Micronet 800 01-837-3699

Bulletin Boards

CBBS (London) 01-399-2136
Forum 80 0482-859169
Mailbox 80 0384-635336
Mailbox 80 (Liverpool) 051-428-8924
TBBS 01-348-9400

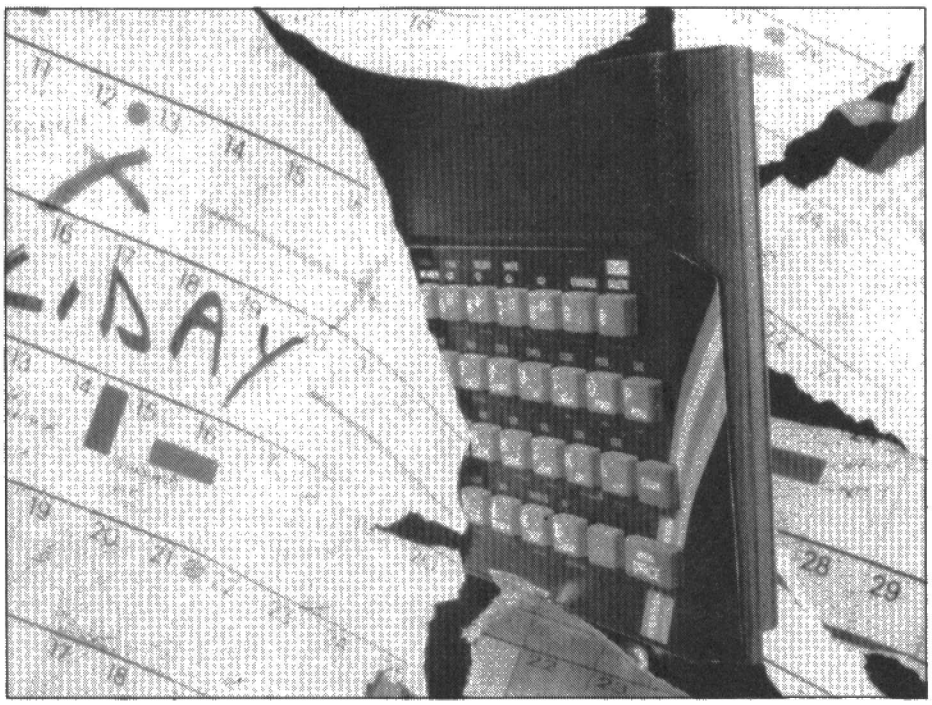
America/Canada Bulletin Boards

ABBS 0101-312-545-8086
Conn-80 0101-212-441-3755
CBBS 0101-312-545-8086
Forum-80 0101-816-861-7040

Manufacturer	Model	Machine/Interface	Price	Baud Rates
Prism	Modem 1000	BBC/Apple IIe	£69.95	1200/75 full duplex 1200/1200 half duplex
Prism	Modem 2000	BBC/Apple IIe	£84.95	1200/75 full duplex 1200/1200 half duplex
Prism	VTX 5000	Spectrum 16/48K	£99.95 & software	1200/75 full duplex 1200/1200 half duplex
Micro-Myte	160 1Q/D	ZX81/Spectrum		Acoustic Coupler sends/receives 100 Baud
Maplin	—	RS232 compatible	£44.95	300/300 full duplex
Tandata	TM 100	BBC/Apple IIe/Com 64	£98.90	1200/75 full duplex
Tandata	TM 200		—	1200/75, 75/1200, 300/300 full 1200/1200 half duplex
DaCom	Buzzbox	RS232 compatible	£79.95	300/300 full duplex
Minor Miracles	WS 2000	RS232 compatible	£99.95	1200/75, 75/1200, 300/300 full 1200/1200 600 half duplex
Display	2B	RS232 compatible	£65.00	300/300 full duplex
Display	Modem 20	RS232	£49.00	1200/75 full duplex
Pace	Grapevine	BBC	£129.95 & software	300/300 full duplex
Watford	Modem 84	RS232	£79.00	1200/75

Spectrum real-time clock

Part 2 has the software to set the clock and PCB foil pattern.



Listing 1. Machine Code.

CLEAR xxxxx where x is your 2K RAM base
Enter machine code at xxxxx plus 2048
Enter xxxxx into 23728
CALL each routine when required.

HEX and DECIMAL code for the "6116 TO SPECTRUM" routine.

11H(17)	78H(120)
00H(0)	77H(119)
08H(8)	01H(1)
2AH(42)	DFH(223)
BOH(176)	FFH(255)
5CH(92)	EDH(237)
01H(1)	78H(120)
DFH(223)	23H(35)
FFH(255)	1BH(27)
EDH(237)	7AH(122)
79H(121)	B3H(179)
01H(1)	CBH(200)
BFH(191)	1BH(24)
FFH(255)	EEH(238)
EDH(237)	

To form the "SPECTRUM TO 6116" routine, replace the three underlined bytes with: 7EH(126), EDH(237), 79H(121).

Listing 2. BASIC.

```
 110 REM Hours minutes seconds display
 119 LET us=IN 62111:REM dummy read
 120 LET us=IN 62111:IF us=15 THEN GO TO 120
 121 LET ts=IN 62367:IF ts=15 THEN GO TO 121
 122 LET um=IN 62623:IF um=15 THEN GO TO 122
 123 LET tm=IN 62879:IF tm=15 THEN GO TO 123
 124 LET uh=IN 63135:IF uh=15 THEN GO TO 124
 125 LET th=IN 63391:IF th=15 THEN GO TO 125
 126 REM Adjust for 12-hour clock
 127 LET a$="AM": LET hrs=th*10+uh: IF hrs>12
     THEN LET hrs=hrs-12: LET a$="PM"
 128 REM Adjust for neat format
 129 IF hrs>9 THEN LET b$="1": LET hrs=hrs-10
     GO TO 131
 130 IF hrs<10 THEN LET b$=" "
 131 PRINT AT 0,0;b$;STR$ hrs;":";STR$ tm;
     STR$ um;":";STR$ ts;STR$ us;" ";a$
 140 GO TO 120
 150 REM ---------------------------------------
 210 REM Lazy method
 219 OUT 65183,0:REM Stop the clock
 220 LET us=IN 62111:LET ts=IN 62367:LET um=
     IN 62623: LET tm=IN 62879:LET uh=IN 63135:
     LET th=IN 63391
 230 PRINT AT 0,0; STR$ th;STR$ uh;":";
     STR$ tm;STR$ um;":";STR$ ts;STR$ us
 240 OUT 65185,15:REM Restart clock
 241 STOP
 250 REM ---------------------------------------
 999 REM Set clock from data statememts
1000 OUT 65185,15: REM Stop the clock
1005 RESTORE 1100
1010 FOR port=64927 TO 62623 STEP -256
1020 READ v: OUT port,v: NEXT port
1030 PAUSE: REM Consult wristwatch then
1035 REM press any key
1040 OUT 65185,0: REM Start clock
1050 STOP
1100 DATA 8: REM leap year
1101 DATA 0,3: REM March
1102 DATA 7: REM day 7=Saturday
1103 DATA 2,6: REM 26th
1104 DATA 1,8: REM 1800hrs
1105 DATA 3,0: REM 30mins
1200 REM ---------------------------------------
1999 REM Load 4 diary bytes
2000 OUT 65503,0: REM Reset counter
2005 RESTORE 2070
2010 FOR n=1 TO 4: REM Four bytes to load
2020 READ v: REM Obtain a byte from data line
2030 OUT 65471,v: REM Load a byte
2040 LET x=IN 65503: REM Increment counter
2050 NEXT n: REM Continue task
2060 STOP
2070 DATA 49,57,56,52: REM 1984 in ASCII
2100 REM ---------------------------------------
2999 REM Recover 4 diary bytes
3000 CLS: OUT 65503,0: REM Reset counter
3010 FOR n=1 TO 4: REM 4 bytes to load
3020 LET v=IN 65471: REM Fetch a byte
3030 PRINT AT 0,n;CHR$ v: REM Print it
3040 LET x=IN 65503: REM Increment counter
3030 NEXT n: REM Continue task
3040 STOP
3999 REM ---------------------------------------
```

A Vero V-Q board is suitable for this project, as it takes the Spectrum connector and the components without undue crowding, but there are just one or two points to bear in mind when building this card. Three of the ICs are CMOS and two of them are expensive, so take extra care with them until the board is fully built. They should, of course, be the last items to go onto the board, and they must go into IC sockets. The best way to bend the pins prior to pushing the chips into their sockets is to lay the IC on a sheet of kitchen-foil and exert the necessary pressure, having first removed any harmful static electricity by grounding yourself and the foil on the nearest water-pipe. The suggested type of NiCad battery solders onto the board, and you may find that a small-bit soldering iron has insufficient heat to do the job. As an alternative, you might consider using a pack of three AA size NiCads in an off-board battery holder. The IC socket SK1 should be the type where the contacts are exposed and not buried deep in the plastic, and it takes two separate "plugs" which are made by cutting up a standard 16-way header. One plug, with 10 contacts, carries the port links, while the other, with 2 contacts, is available for the INTERRUPT signals. The board will be that much easier to fit onto the Spectrum edge connector if the 28 x 28-way socket is mounted at the very bottom of the V-Q board, standing 10mm proud of the board's surface.

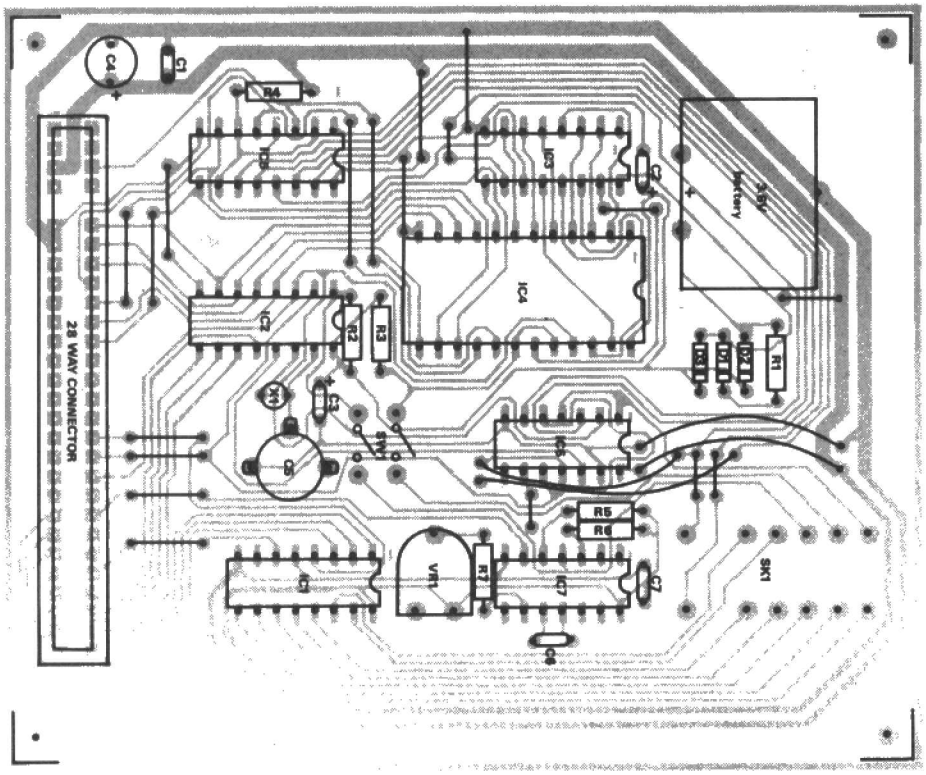


Figure 1(a). Pin-outs and overlay.

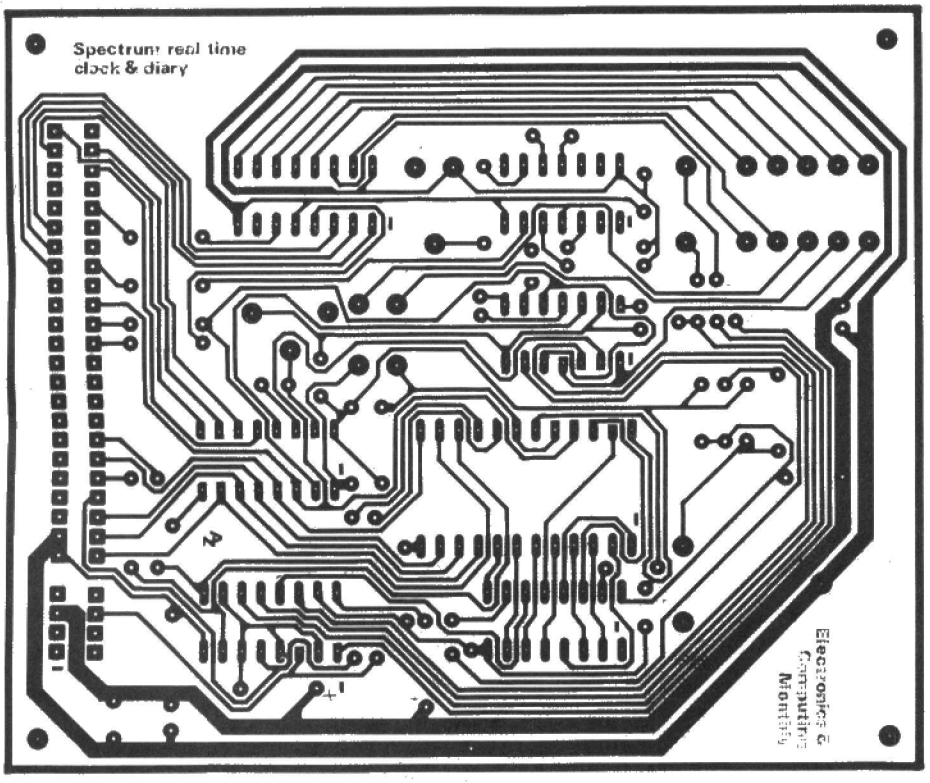


Figure 1(b). PCB foil pattern.

Software

The accompanying BASIC listings will allow simple experimentation with the timer IC. The periodic pulses available from the interrupt pin of the IC may prove useful to constructors, while the use of mode 2 interrupt within the Spectrum itself should allow machine-coders to read the clock every 20mS and display the time constantly on the screen. As far as the BASIC is concerned, RUN 1000 will set the clock and RUN 110 or RUN 210 will read a simple time value. The diary RAM can also be used from BASIC although if large amounts of data are being moved between CMOS RAM and Spectrum RAM then a machine code utility ought to be used. RUN 2000 loads the year into the first four locations of the diary, while RUN 3000 reads them back again. The machine code programs are relocatable: the first 29 bytes shifts 2K of Spectrum data to CMOS RAM, while the second 29 bytes shifts data in the opposite direction. In each case the starting address of the desired Spectrum RAM base is loaded into address 23728 and 23729, low byte first. ■

PARTS LIST

Integrated circuits	
IC1	74LS138
IC2	MM58174
IC3	4040 BE
IC4	6116
IC5	74LS368
IC6	74LS32
IC7	74121
Resistors	
R1, R2	19k
R3, R4, R5, R6	1k
R7	2k
VR8	10k
Capacitors	
C1, C7	100n
C2, C3	10u
C4	100u
C5	5-60p
C6	100p
Diodes	
D1-3	1N4148
Crystal	
X1	32.768kHz
Miscellaneous	
NiCad 3.6V; SW1, 2-pole on/off sub-min; IC sockets; One 16-pin header; 28 x 28 Spectrum socket.	

On screen multimeter

A program for the Hi-Res graphics board and the analogue board of the ECM Hi-Res Computer.

By Alan Stirling and Paul Izod.

Over the past few months we have been presenting new hardware for this system, based on the Motorola MC6809 processor. Now it is time to demonstrate what the system can do when given a suitable application.

With the recent introduction of the analogue board, offering 8 channels of A-D conversion with 12-bit accuracy, such an application evolved. This was the design of an 8 channel multi-meter offering an input voltage range of −10 Volts to +10 Volts using all the input channels of the A-D board with a suitable display on the Hi-Res graphics board. The final design of the display provides a digital display of the input voltage, together with a moving bar representing its analogue magnitude. Positive voltages are displayed in red, negative in blue. The powerful instruction set of the 6809 allows the programmer to support this hardware with very efficient code, thus providing an elegant solution for this application.

The fully annotated listing below provides an example to all 6809 programmers of the power and flexibility of this processor.

Hardware specification

The 12-bit A-D converter offers an accuracy of greater than 0.01 Volts, so the digital display provides a three digit readout with a range from −9.99 to −0.01 to 0.00 to +0.01 to +9.99. Positive values are displayed without the sign.

Depending upon the clock speed of the processor, the program reads and displays all channels at a rate of about 2 complete cycles per second.

The bar display has a horizontal resolution of about 200 pixels, offering a displayed resolution of about 0.05 Volts.

Software specification

The program has been written to be as independant as possible, having been tested on a number of differing ECM Hi-Res Computer Systems. The version presented here is adapted for use with the disk operating system, FLEX, although it could easily be converted to load from cassette and run on a system using system monitor routines instead of calls to FLEX routines.

"the multimeter listing provides an example to all 6809 programmers of the power of this processor".

All addresses used are defined at the start of the program where they can be easily identified and changed if necessary.

The program has been written in a modular fashion, so as to make it easy to follow and easy to modify, and it has been fully annotated for the same reason. The 6809 is perfectly adapted for use in this application, since its indexed addressing and hardware multiply facilities can be fully utilised. There are few, if any, 8-bit processors that can rival the 6809 in this type of operation.

Program Overview

The program is split into logical parts in the following order:

- System Equates – Hardware & Software Addresses
- Variable Storage Areas & Parameter Tables
- Subroutine Area
- Main Program Loop
- Main Routines called by Main Program Loop

System Equates. These program statements define the position within the machine of particular software routines and hardware registers. These can be altered if your system does not provide these facilities at these particular addresses.

System Variables. Any values that the program needs to control are held here, with the parameter table at the end. This table controls the whole operation of the program, since it holds vital information for each channel of the display. In detail, these are as follows:

X-Axis Address of Start of Display Box on Screen
Y-Axis Address of Start of Display Box on Screen
Memory Address of Digital Output from Analogue Channel
Text Description of Channel (9 Characters)

These parameters can be modified for each channel, allowing the order of display, the position on the screen, the channel to be used and the name given to each channel, to be amended.

Main Program Loop. It is from within this loop that all other routines are called. Arranging the program in this way ensures that it is easy to understand and modify.

Apart from reading the digital input, converting the result to a binary-coded decimal number and displaying this number in both digital and analogue forms, this main loop causes the pointer to the parameter table to step through all 8 inputs, and monitors the system keyboard for input, which causes it to stop operation and return to FLEX.

Finally it also monitors the least significant bit of the digital input byte on the analogue board. If this bit is set low, the routine halts, holding the last values displayed. On releasing this bit, the program continues. This feature can be used to 'latch' results on the screen.

Subroutine Area. The main routine here is called 'Block'. This is used to draw a coloured rectangle on the screen. It is drawn with its lower left corner at the position of the cursor, with its size held in the X register, the top 8 bits being the X-axis dimension, the lower 8 bits being for the Y-axis. The colour is transferred in the A register.

Other subroutines are used to move the cursor and test that the graphics board is ready for the next command.

Main Routines. It is in these routines that initially the screen display is set up and the analogue board is triggered.

Once this has been completed, each input is read sequencially, the value being

Listing 1: Multimeter

```
                        ****************************************
                        **   PROGRAM NAME - MULTI-METER       **
                        **                                    **
                        **   FUNCTION - 8 DIGITAL & ANALOGUE  **
                        **   DISPLAYS ON THE HI-RES GRAPHICS  **
                        **   BOARD WITH CONTINUOUS MONITORING **
                        **                                    **
                        **   HARDWARE REQD - 6809 CPU, 64K,   **
                        **   ANALOGUE, HI-RES GRAPHICS, DISK  **
                        **                                    **
                        **   WRITTEN BY ALAN STIRLING 03/12/83 **
                        ****************************************
                        *
                        * START OF FLEX EQUATES
                        *
              CD03  WARMS   EQU   $CD03     FLEX WARM START
              CD15  GETCHR  EQU   $CD15     FLEX GET A CHAR
              CD18  PUTCHR  EQU   $CD18     FLEX SEND A CHAR
              CD4E  STAT    EQU   $CD4E     FLEX CHECK INPUT STATUS
                        *
                        * START OF GENERAL EQUATES
                        *
              F100  VCMD    EQU   $F100     VIDEO COMMAND REG
              F105  VDX     EQU   $F105     VIDEO DELTA X REG
              F107  VDY     EQU   $F107     VIDEO DELTA Y REG
              F108  VX      EQU   $F108     VIDEO X REG
              F10A  VY      EQU   $F10A     VIDEO Y REG
              F120  VCOL    EQU   $F120     VIDEO COLOUR REG
              F140  VKEY    EQU   $F140     VIDEO KEYBOARD INPUT
              F160  VBORD   EQU   $F160     VIDEO BORDER REG
              F200  ACMD    EQU   $F200     ANALOGUE COMMAND REG
              F207  ATRG    EQU   $F207     ANALOGUE TRIGGER REG
              F280  ADATA   EQU   $F280     ANALOGUE DATA REGS
                        *
C100                    ORG   $C100     UTILITY COMMAND SPACE
                        *
C100 20  0D     START   BRA   START1
                        *
C102 01         VN      FCB   1          VERSION NUMBER
C103 01E8       FACTOR  FDB   488        mV PER DIGIT
C105 D2         BARFCT  FCB   $D2        FULL SCALE BAR LENGTH
C106            COUNT   RMB   1          METER NUMBER
C107            SIGN    RMB   1          SIGN FLAG
C108            BCOUNT  RMB   1          BINARY COUNT
C109            VALUE   RMB   2          INPUT VALUE
C10B            HOLD    RMB   4          MULTIPLY RESULT
                        *
C10F 16  013D   START1  LBRA  START2
                        *
C112 56 02 00 00 SETUP  FCB   $56,$02,$00,$00
C116            BIN     RMB   4
C11A            DEC     RMB   4
                        *
                        * PARAMETER TABLE
                        *
                        * X-AXIS ADDR, Y-AXIS ADDR, INPUT ADDR
                        *
C11E 0008 0188  TABLE   FDB   $0008,$0188,ADATA
C122 F280
C124 43 48 41 4E        FCC   /CHANNEL 1/
C128 4E 45 4C 20
C12C 31
C12D 0108 0188          FDB   $0108,$0188,ADATA+2
C131 F282
C133 43 48 41 4E        FCC   /CHANNEL 2/
C137 4E 45 4C 20
C13B 32
C13C 0008 0108          FDB   $0008,$0108,ADATA+4
C140 F284
C142 43 48 41 4E        FCC   /CHANNEL 3/
C146 4E 45 4C 20
C14A 33
C14B 0108 0108          FDB   $0108,$0108,ADATA+6
C14F F286
C151 43 48 41 4E        FCC   /CHANNEL 4/
C155 4E 45 4C 20
C159 34
C15A 0008 0088          FDB   $0008,$0088,ADATA+8
C15E F288
C160 43 48 41 4E        FCC   /CHANNEL 5/
C164 4E 45 4C 20
C168 35
C169 0108 0088          FDB   $0108,$0088,ADATA+10
C16D F28A
C16F 43 48 41 4E        FCC   /CHANNEL 6/
C173 4E 45 4C 20
C177 36
C178 0008 0008          FDB   $0008,$0008,ADATA+12
C17C F28C
C17E 43 48 41 4E        FCC   /CHANNEL 7/
C182 4E 45 4C 20
C186 37
C187 0108 0008          FDB   $0108,$0008,ADATA+14
C18B F28E
C18D 43 48 41 4E        FCC   /CHANNEL 8/
C191 4E 45 4C 20
C195 38
                        *
                        ***** SUBROUTINE AREA
                        *
                        * WRITE BLOCK TO SCREEN
                        *
C196 8D  63     BLOCK   BSR   TEST       IS VIDEO READY ?
C198 84  07             ANDA  #$07       SELECT COLOUR ONLY
C19A B7  F120           STA   VCOL       SAVE COLOUR
C19D 1F  10             TFR   X,D        MOVE X & Y REG INTO A & B
C19F B7  F105           STA   VDX        STORE DELTA X VALUE
C1A2 F7  F107           STB   VDY        STORE DELTA Y VALUE
                        *
C1A5 8D  54     BLUP    BSR   TEST
C1A7 C6  12             LDB   #$12       MOVE UP VECTOR COMMAND
C1A9 F7  F100           STB   VCMD       DO IT
C1AC 8D  4D             BSR   TEST
C1AE F6  F101           LDB   VCMD+1     PICK UP CTRL1 REG
C1B1 34  04             PSHS  B          SAVE IT
C1B3 5F                 CLRB             ZEROIZE B REG
C1B4 F7  F101           STB   VCMD+1     SET TO SKIP
C1B7 8D  42             BSR   TEST
C1B9 C6  14             LDB   #$14       MOVE DOWN VECTOR COMMAND
C1BB F7  F100           STB   VCMD       DO IT
C1BE 8D  3B             BSR   TEST
C1C0 35  04             PULS  B          RESTORE B REG
C1C2 F7  F101           STB   VCMD+1     RESTORE CTRL1 REG
C1C5 4D                 TSTA             REACHED X-AXIS COUNT
C1C6 26  0B             BNE   BLRT       FINISHED?
C1C8 8D  31             BSR   TEST
C1CA 1F  10             TFR   X,D        GET X AGAIN
C1CC C6  16             LDB   #$16       MOVE LEFT COMMAND
C1CE F7  F100           STB   VCMD       SEND IT
C1D1 20  28             BRA   TEST       WAIT TILL FINISHED
                        *
C1D3 8D  26     BLRT    BSR   TEST
C1D5 C6  A0             LDB   #$A0       MOVE LEFT SMALL VECTOR
C1D7 F7  F100           STB   VCMD       DO IT
C1DA 4A                 DECA             REDUCE X-AXIS COUNT
C1DB 20  C8             BRA   BLUP       REPEAT
                        *
                        * MOVE GRAPHICS CURSOR
                        *
C1DD 8D  1C     MOVE    BSR   TEST
C1DF 5F                 CLRB
C1E0 F7  F101           STB   VCMD+1     SET TO SKIP
C1E3 8D  16             BSR   TEST
C1E5 1E  10             EXG   X,D        MOVE X & Y TO A & B
C1E7 B7  F105           STA   VDX        SAVE DELTA X VALUE
C1EA F7  F107           STB   VDY        SAVE DELTA Y VALUE
C1ED 1E  10             EXG   X,D        RESTORE A REG
C1EF B7  F100           STA   VCMD       ACTION VECTOR
C1F2 8D  07             BSR   TEST
C1F4 C6  01             LDB   #$01
C1F6 F7  F101           STB   VCMD+1     SET TO DRAW
C1F9 20  00             BRA   TEST       WAIT TILL FINISHED
                        *
                        * TEST FOR GRAPHICS BOARD READY
                        *
C1FB F6  F100   TEST    LDB   VCMD       VIDEO STATUS
C1FE C5  04             BITB  #$04       TEST BUSY
C200 27  F9             BEQ   TEST       YES, REPEAT TEST
C202 39                 RTS              NO, EXIT
                        *
                        * SET UP POSITION & COLOURS FOR VALUES
                        *
>C203 17  FFF5  SETDIG  LBSR  TEST
 C206 EC  A4            LDD   0,Y        PICK UP X-AXIS POINTER
 C208 C3  0090          ADDD  #$0090     ADD IN OFFSET
 C20B FD  F108          STD   VX         SET UP X REGISTER
>C20E 17  FFEA          LBSR  TEST
 C211 EC  22            LDD   2,Y        PICK UP Y-AXIS POINTER
 C213 C3  0007          ADDD  #$0007     ADD IN OFFSET
 C216 FD  F10A          STD   VY         SET UP Y REGISTER
>C219 17  FFDF          LBSR  TEST
 C21C 86  33            LDA   #$33       SET CHARACTER SIZE
 C21E B7  F103          STA   VCMD+3     SAVE IT
>C221 17  FFD7          LBSR  TEST
 C224 86  06            LDA   #$06       GREEN ON YELLOW GIVES RED
 C226 B7  F120          STA   VCOL       SET COLOUR
>C229 17  FFCF          LBSR  TEST
 C22C 39                RTS              EXIT
                        *
                        * SET UP POSITION FOR BAR
                        *
>C22D 17  FFCB  SETBAR  LBSR  TEST
 C230 EC  A4            LDD   0,Y        PICK UP X-AXIS POINTER
 C232 C3  000D          ADDD  #$000D     ADD IN OFFSET
 C235 FD  F108          STD   VX         SET UP X REGISTER
>C238 17  FFC0          LBSR  TEST
 C23B EC  22            LDD   2,Y        PICK UP Y-AXIS POINTER
 C23D C3  004A          ADDD  #$004A     ADD IN OFFSET
 C240 FD  F10A          STD   VY         SET UP Y REGISTER
>C243 17  FFB5          LBSR  TEST
 C246 39                RTS              EXIT
                        *
                        * TEST FOR BUTTON HELD DOWN
                        *
 C247 B6  F200  STOP    LDA   ACMD       LOAD BYTE FOR TEST
 C24A 85  01            BITA  #$01       TEST LSB
 C24C 27  F9            BEQ   STOP       IF ZERO, REPEAT
 C24E 39                RTS              ELSE EXIT BACK TO MAIN LOOP
                        *
                        ***** MAIN PROGRAM LOOP
                        *
>C24F 17  003A  START2  LBSR  CLSCN      CLEAR SCREEN ROUTINE
>C252 17  0053          LBSR  SETANG     SET UP ANALOGUE ROUTINE
>C255 17  0059          LBSR  SETDIS     SET UP DISPLAY ROUTINE
                        *
 C258 17  014D  START3  LBSR  SETVAL     SET UP LOOP CONTROLS
 C25B 17  015A  REPEAT  LBSR  READVL     READ VALUE ROUTINE
 C25E 17  01AE          LBSR  CALCVL     CALCULATE ANALOGUE VALUE
 C261 17  01EE          LBSR  REMOVL     REMOVE OLD VALUE
 C264 17  01FE          LBSR  DISPVL     DISPLAY VALUE ROUTINE
 C267 17  0248          LBSR  CALCBR     CALCULATE LENGTH OF BAR
 C26A 17  0263          LBSR  REMOBR     REMOVE OLD BAR
 C26D 17  026B          LBSR  DISPBR     DISPLAY NEW BAR
>C270 17  FFD4          LBSR  STOP       TEST FOR BUTTON HELD
 C273 17  0AD8          LBSR  STAT       TEST FOR END OF RUN
>C276 1026 000A         LBNE  EXIT       YES, JUMP BACK TO FLEX
 C27A 108C C196         CMPY  #BLOCK     REACHED END OT TABLE
>C27E 1026 FFD9         LBNE  REPEAT     NO, REPEAT
 C282 20  D4            BRA   START3     BACK TO FIRST INPUT
                        * EXIT BACK TO FLEX
                        *
 C284 86  0C    EXIT    LDA   #$0C       CLEAR PAGE COMMAND
 C286 BD  CD18          JSR   PUTCHR     SEND CHARACTER
 C289 7E  CD03          JMP   WARMS      EXIT TO FLEX
                        * CLEAR SCREEN
                        *
 C28C 7F  F120  CLSCN   CLR   VCOL       CLEAR COLOUR REGISTER
 C28F 7F  F160          CLR   VBORD      CLEAR BORDER REGISTER
 C292 17  FF66          LBSR  TEST       WAIT TILL READY
 C295 86  01            LDA   #$01       SET FOR WRITE
 C297 B7  F101          STA   VCMD+1     STORE IN VCTRL1 REG
 C29A 86  07            LDA   #$07       CLEAR SCREEN COMMAND
 C29C B7  F100          STA   VCMD       SEND IT
 C29F 17  FF59          LBSR  TEST       WAIT TILL FINISHED
 C2A2 86  01            LDA   #$01       RESTORE WRITE COMMAND
 C2A4 B7  F101          STA   VCMD+1     STORE IN VCTRL1 REG
 C2A7 39                RTS              EXIT BACK TO MAIN LOOP
                        * SET UP ANALOGUE BOARD
                        *
 C2A8 86  57    SETANG  LDA   #$57       SCAN ALL 8 CHANNELS
 C2AA B7  F200          STA   ACMD       SET UP REGISTER
 C2AD B7  F207          STA   ATRG       SET INTERNAL TRIGGER
 C2B0 39                RTS              EXIT TO MAIN LOOP
```

Multimeter

```
                        * SET UP DISPLAY SCREEN
                        *
C2B1 108E C11E  SETDIS  LDY    #TABLE    POINT AT PARAMETERS
C2B5 17   FF43  LOOP    LBSR   TEST      FINISHED LAST COMMAND ?
C2B8 AE   A1            LDX    0,Y++     LOAD X PARAMETER
C2BA BF   F108          STX    VX        STORE IT
C2BD AE   A4            LDX    0,Y       LOAD Y PARAMETER
C2BF BF   F10A          STX    VY        STORE IT
C2C2 31   24            LEAY   4,Y       SKIP OVER INPUT ADDRESS
C2C4 8E   F070          LDX    #$F070    SIZE PARAMETER
C2C7 86   01            LDA    #$01      BLUE
C2C9 17   FECA          LBSR   BLOCK
C2CC C6   D1            LDB    #$D1      MOVE 2 UP & RIGHT
C2CE F7   F100          STB    VCMD      SEND IT
C2D1 86   02            LDA    #$02      GREEN
C2D3 8E   EC22          LDX    #$EC22    DELTA X/Y VALUES
C2D6 17   FEBD          LBSR   BLOCK     SEND IT
C2D9 8E   0028          LDX    #$0028    DELTA X/Y VALUES
C2DC 86   12            LDA    #$12      UP COMMAND
C2DE 17   FEFC          LBSR   MOVE
C2E1 8E   EC44          LDX    #$EC44    DELTA X/Y VALUES
C2E4 86   02            LDA    #$02      GREEN
C2E6 17   FEAD          LBSR   BLOCK     SEND IT
C2E9 8E   0814          LDX    #$0814
C2EC 86   11            LDA    #$11      UP RIGHT
C2EE 17   FEEC          LBSR   MOVE
C2F1 8E   DC2C          LDX    #$DC2C
C2F4 86   01            LDA    #$01      BLUE
C2F6 17   FE9D          LBSR   BLOCK
C2F9 8E   0202          LDX    #$0202
C2FC 86   11            LDA    #$11      UP RIGHT
C2FE 17   FEDC          LBSR   MOVE
C301 8E   D828          LDX    #$D828
C304 86   07            LDA    #$07
C306 17   FE8D          LBSR   BLOCK
C309 8E   0114          LDX    #$0114
C30C 86   15            LDA    #$15      DOWN VECTOR
C30E 17   FECC          LBSR   MOVE
C311 86   00            LDA    #$00
C313 B7   F120          STA    VCOL      SET COLOUR
C316 86   30            LDA    #$30      SET TO '0'
C318 C6   12            LDB    #$12      SELECT SIZE OF CHAR
C31A F7   F103          STB    VCMD+3    SAVE IN CHAR SIZE REG
C31D 17   FEDB  NEXT    LBSR   TEST
C320 B7   F100          STA    VCMD      OUTPUT CHAR
C323 34   02            PSHS   A         SAVE COUNT
C325 17   FED3          LBSR   TEST
C328 8E   0614          LDX    #$0614    DELTA X/Y VALUE
C32B 86   13            LDA    #$13
C32D 17   FEAD          LBSR   MOVE
C330 8E   0408          LDX    #$0408
C333 86   01            LDA    #$01      BLUE
C335 17   FE5E          LBSR   BLOCK
C338 8E   1514          LDX    #$1514
C33B 86   15            LDA    #$15      DOWN RIGHT COMMAND
C33D 17   FE9D          LBSR   MOVE
C340 86   00            LDA    #$00      BLACK
C342 B7   F120          STA    VCOL      SET COLOUR
C345 35   02            PULS   A         RESTORE COUNT
C347 4C                 INCA             SELECT NEXT NUMBER
C348 81   3A            CMPA   #$3A      REACHED '9' ?
C34A 26   D1            BNE    NEXT      NO, DISPLAY NEXT NUMBER
C34C 8E   0300          LDX    #$0300    DELTA X/Y VALUE
C34F 86   16            LDA    #$16      MOVE LEFT
C351 17   FE89          LBSR   MOVE
C354 86   31            LDA    #$31      LOAD '1'
C356 17   FEA2          LBSR   TEST
C359 B7   F100          STA    VCMD      OUTPUT CHAR
C35C 86   30            LDA    #$30      LOAD '0'
C35E 17   FE9A          LBSR   TEST
C361 B7   F100          STA    VCMD      OUTPUT CHAR
C364 8E   0914          LDX    #$0914    DELTA X/Y VALUE
C367 86   13            LDA    #$13      UP LEFT VECTOR
C369 17   FE71          LBSR   MOVE
C36C 8E   0408          LDX    #$0408
C36F 86   01            LDA    #$01      BLUE
C371 17   FE22          LBSR   BLOCK
C374 8E   D339          LDX    #$D339    DELTA X/Y VALUE
C377 86   17            LDA    #$17      DOWN LEFT COMMAND
C379 17   FE61          LBSR   MOVE
C37C 86   23            LDA    #$23      CHAR SIZE
C37E B7   F103          STA    VCMD+3    SAVE IT
C381 17   FE77          LBSR   TEST
C384 86   05            LDA    #$05      MAGENTA
C386 B7   F120          STA    VCOL      SAVE IN COLOUR REG
C389 86   09            LDA    #$09      LOAD COUNT
C38B B7   C106          STA    COUNT     SAVE IT
C38E A6   A4    AGAIN   LDA    0,Y       LOAD CHAR
C390 17   FE68          LBSR   TEST
C393 B7   F100          STA    VCMD      OUTPUT CHAR
C396 86   20            LDA    #$20      SPACE CHARACTER
C398 A7   A0            STA    0,Y+      CLEAR TABLE
C39A 7A   C106          DEC    COUNT     ANY MORE CHARACTERS?
C39D 26   EF            BNE    AGAIN     YES, GET NEXT
C39F 108C C196          CMPY   #BLOCK    REPEAT ?
C3A3 1026 FF0E          LBNE   LOOP      YES
C3A7 39                 RTS              EXIT BACK TO MAIN LOOP
                        *
                        * SET UP READ & DISPLAY LOOPS
                        *
C3A8 108E C11E  SETVAL  LDY    #TABLE    POINT AT FIRST SCREEN POSN
C3AC 17   FE4C          LBSR   TEST
C3AF 86   03            LDA    #$03      SET FOR READ-MODIFY-WRITE
C3B1 B7   F101          STA    VCMD+1    SAVE IT
C3B4 17   FE44          LBSR   TEST
C3B7 39                 RTS              EXIT BACK TO MAIN LOOP
                        *
                        * READ ANALOGUE INPUT VALUE
                        *
C3B8 7F   C107  READVL  CLR    SIGN      RESET SIGN FLAG
C3BB EC   B8 04         LDD    [4,Y]     LOAD ANALOGUE VALUE
C3BE 85   08            BITA   #$08      TEST SIGN BIT
C3C0 27   04            BEQ    NEGA      SKIP IF NEGATIVE
C3C2 84   07            ANDA   #$07      REMOVE TOP BITS
C3C4 20   0C            BRA    POST      SKIP
C3C6 7C   C107  NEGA    INC    SIGN      SET SIGN FLAG
C3C9 40                 NEGA             CONVERT TO 2'S COMPLIMENT
C3CA 50                 NEGB             CONVERT TO 2'S COMPLIMENT
C3CB 82   00            SBCA   #0        USE BORROW IF ANY
C3CD C3   0800          ADDD   #$800     CONVERT TO NEG GOING NUMBER
C3D0 84   0F            ANDA   #$0F      STRIP OFF TOP 4 BITS
C3D2 FD   C109  POST    STD    VALUE     SAVE 11 BIT VALUE
C3D5 B6   C104          LDA    FACTOR+1  LOAD mV PER DIGIT(LSB)
C3D8 F6   C10A          LDB    VALUE+1   LOAD VALUE(LSB)
C3DB 3D                 MUL              MULTIPLY LSB'S
C3DC FD   C10D          STD    HOLD+2    SAVE PARTIAL PRODUCT
C3DF B6   C104          LDA    FACTOR+1  LOAD mV PER DIGIT(LSB)
C3E2 F6   C109          LDB    VALUE     LOAD VALUE(MSB)
C3E5 3D                 MUL              MULTIPLY LSB & MSB
C3E6 FB   C10D          ADDB   HOLD+2    ADD IN PARTIAL PRODUCT
C3E9 89   00            ADCA   #0        ADD CARRY IF ANY
C3EB FD   C10C          STD    HOLD+1    SAVE SUM OF PARTIAL PRODUCTS
                        *
                        * USE MSB OF FACTOR
                        *
C3EE B6   C103          LDA    FACTOR    LOAD mV PER DIGIT(MSB)
C3F1 F6   C10A          LDB    VALUE+1   LOAD VALUE(LSB)
C3F4 3D                 MUL              MULTIPLY MSB & LSB
C3F5 F3   C10C          ADDD   HOLD+1    ADD IN PREVIOUS PRODUCT
C3F8 FD   C10C          STD    HOLD+1    SAVE PARTIAL PRODUCTS
C3FB 4F                 CLRA             PREPARE FOR CARRY TEST
C3FC 89   00            ADCA   #0        ADD IN CARRY, IF ANY
C3FE B7   C10B          STA    HOLD      SAVE TOP BYTE
C401 B6   C103          LDA    FACTOR    LOAD mV PER DIGIT(MSB)
C404 F6   C109          LDB    VALUE     LOAD VALUE(MSB)
C407 3D                 MUL              MULTIPLY MSB'S
C408 F3   C10B          ADDD   HOLD      ADD IN PARTIAL PRODUCT
C40B FD   C10B          STD    HOLD      SAVE FINAL SUM OF PRODUCTS
C40E 39                 RTS              EXIT BACK TO MAIN LOOP
                        *
                        * CALCULATE ANALOGUE VALUE
                        *
C40F 8E   C116  CALCVL  LDX    #BIN      POINT AT START VALUES
C412 6F   04    BMOVE   CLR    4,X       CLEAR DEC VALUES
C414 A6   1C            LDA    -4,X      PICK UP START VALUE
C416 A7   80            STA    0,X+      SAVE IN BINARY VALUE
C418 8C   C11A          CMPX   #DEC      REACHED END OF TABLE ?
C41B 26   F5            BNE    BMOVE     NO, TRANSFER NEXT VALUE
C41D 86   12            LDA    #$12      LOAD BIT COUNT
C41F B7   C108          STA    BCOUNT    SET UP COUNTER
C422 74   C10C  BAGAIN  LSR    HOLD+1    MOVE DOWN
C425 76   C10D          ROR    HOLD+2    SELECT BOTTOM BIT IN C
C428 8E   C116          LDX    #BIN      POINT AT BINARY VALUES
C42B 24   11            BCC    BINCR     IF CLEAR, SKIP
C42D C6   04            LDB    #$04      POINT AT DECIMAL VALUES
C42F 1C   FE            CLC              CLEAR CARRY
C431 A6   80    DOWN    LDA    0,X+      GET TWO DIGITS OF FIRST NO
C433 A9   03            ADCA   3,X       ADD TWO DIGITS OF SECOND NO
C435 19                 DAA              ADJUST RESULT
C436 A7   03            STA    3,X       STORE RESULT
C438 5A                 DECB             REDUCE COUNT
C439 26   F6            BNE    DOWN      NO, MOVE TO NEXT
C43B 8E   C116          LDX    #BIN      POINT AT BINARY VALUES
C43E C6   04    BINCR   LDB    #$04      POINT AT DECIMAL VALUES
C440 1C   FE            CLC              CLEAR CARRY
C442 A6   84    DOWO    LDA    0,X       GET TWO DIGITS OF FIRST NO
C444 A9   84            ADCA   0,X       ADD TWO DIGITS OF SECOND NO
C446 19                 DAA              ADJUST RESULT
C447 A7   80            STA    0,X+      STORE RESULT
C449 5A                 DECB             REDUCE COUNT
C44A 26   F6            BNE    DOWO      NO, MOVE TO NEXT
C44C 7A   C108          DEC    BCOUNT    REDUCE COUNT
C44F 26   D1            BNE    BAGAIN    NON-ZERO, REPEAT
C451 39                 RTS              EXIT BACK TO MAIN LOOP
                        * REMOVE LAST VALUE FROM SCREEN
                        *
C452 17   FDAE  REMOVL  LBSR   SETDIG
C455 86   06            LDA    #$06      COUNT
C457 17   FDA1  REREVL  LBSR   TEST
C45A E6   A6            LDB    A,Y       PICK UP CHAR
C45C F7   F100          STB    VCMD      SEND TO SCREEN
C45F 4C                 INCA             MOVE TO NEXT CHAR
C460 81   0B            CMPA   #$0B      END OF TABLE ?
C462 26   F3            BNE    REREVL    NO, GET NEXT CHAR
C464 39                 RTS              EXIT BACK TO MAIN ROUTINE
                        * DISPLAY ANALOGUE VALUES
                        *
C465 17   FD9B  DISPVL  LBSR   SETDIG
C468 86   20            LDA    #$20      LOAD SPACE
C46A 7D   C107          TST    SIGN      TEST SIGN FLAG
C46D 27   02            BEQ    POSIT     IF ZERO, POSITIVE
C46F 86   2D            LDA    #'-       ELSE LOAD MINUS SIGN
C471 A7   26    POSIT   STA    6,Y       SAVE IN TABLE
C473 B7   F100          STA    VCMD
C476 17   FD82          LBSR   TEST
C479 B6   C11C          LDA    DEC+2     LOAD DECIMAL VALUE
C47C 44                 LSRA
C47D 44                 LSRA
C47E 44                 LSRA             MOVE TO LSB
C47F 44                 LSRA
C480 8B   30            ADDA   #$30      CONVERT TO ASCII
C482 A7   27            STA    7,Y       SAVE IN TABLE
C484 B7   F100          STA    VCMD      SEND IT
C487 17   FD71          LBSR   TEST
C48A 86   2E            LDA    #'.       LOAD '.'
C48C A7   28            STA    8,Y       SAVE IN TABLE
C48E B7   F100          STA    VCMD      SEND IT
C491 17   FD67          LBSR   TEST
C494 B6   C11C          LDA    DEC+2     LOAD DECIMAL VALUE
C497 84   0F            ANDA   #$0F      SELECT LSB
C499 8B   30            ADDA   #$30      CONVERT TO ASCII
C49B A7   29            STA    9,Y       SAVE IN TABLE
C49D B7   F100          STA    VCMD      SEND IT
C4A0 17   FD58          LBSR   TEST
C4A3 B6   C11B          LDA    DEC+1     LOAD DECIMAL VALUE
C4A6 44                 LSRA
C4A7 44                 LSRA
C4A8 44                 LSRA             MOVE TO LSB
C4A9 44                 LSRA
C4AA 8B   30            ADDA   #$30      CONVERT TO ASCII
C4AC A7   2A            STA    10,Y      SAVE IN TABLE
C4AE B7   F100          STA    VCMD      SEND IT
C4B1 39                 RTS              EXIT BACK TO MAIN LOOP
                        * CALCULATE BAR LENGTH
                        *
C4B2 FC   C109  CALCBR  LDD    VALUE     LOAD ANALOGUE VALUE
C4B5 81   08            CMPA   #$08      OVER FULL SCALE DEFLECTION ?
C4B7 26   03            BNE    NOTFSD
C4B9 CC   07FF          LDD    #$7FF     SET TO FULL SCALE DEFLECTION
```

Multimeter

```
C4BC 58          NOTFSD  LSLB
C4BD 49                  ROLA
C4BE 58                  LSLB
C4BF 49                  ROLA
C4C0 58                  LSLB
C4C1 49                  ROLA
C4C2 58                  LSLB
C4C3 49                  ROLA
C4C4 58                  LSLB
C4C5 49                  ROLA                    GIVES 8 BIT VALUE IN A
C4C6 F6  C105            LDB     BARFCT          LOAD BAR LENGTH FACTOR
C4C9 3D                  MUL
C4CA 8B  02              ADDA    #$02            OFFSET RIGHT BY 2 PIXELS
C4CC B7  C11A            STA     DEC             HOLD BAR LENGTH
C4CF 39                  RTS                     EXIT BACK TO MAIN LOOP
                 *
                 * REMOVE OLD BAR
                 *
C4D0 17  FD5A    REMOBR  LBSR    SETBAR          POSITION GRAPHICS CURSOR
C4D3 A6  2B              LDA     11,Y            LOAD COLOUR
C4D5 AE  2C              LDX     12,Y            LOAD BLOCK SIZE
C4D7 17  FCBC            LBSR    BLOCK           REMOVE OLD BLOCK
C4DA 39                  RTS                     EXIT BACK TO MAIN LOOP
                 *
                 * DISPLAY NEW BAR
                 *
C4DB 17  FD4F    DISPBR  LBSR    SETBAR          POSITION GRAPHICS CURSOR
C4DE 86  03              LDA     #$03            POSITIVE COLOUR
C4E0 7D  C107            TST     SIGN            CHECK SIGN FLAG
C4E3 27  02              BEQ     POSTV           IF ZERO, SKIP
C4E5 86  04              LDA     #$04            NEGATIVE BAR COLOUR
C4E7 A7  2B      POSTV   STA     11,Y            SAVE COLOUR
C4E9 34  02              PSHS    A               SAVE COLOUR
C4EB B6  C11A            LDA     DEC             LOAD BAR LENGTH
C4EE C6  1C              LDB     #$1C            BAR WIDTH
C4F0 ED  2C              STD     12,Y            SAVE BLOCK SIZE
C4F2 1F  01              TFR     D,X             MOVE TO X REG
C4F4 35  02              PULS    A               RESTORE COLOUR
C4F6 17  FC9D            LBSR    BLOCK           DRAW BAR
C4F9 31  2F              LEAY    15,Y            MOVE TO START OF NEXT VALUES
C4FB 39                  RTS                     EXIT BACK TO MAIN LOOP
                 *
```

converted to binary milli-Volts, using a 16-bit multiply routine and then converted again to a three digit binary-coded decimal value for display.

A further 8-bit multiply is required to produce the correct bar length. Finally the old bar is removed, using the read-modify-write mode, where if the same shape and colour is written twice, the second write will remove all trace of the first. The original values of each bar are stored in the parameter table whilst other channels are being displayed. The new bar is then written to the screen, and its parameters stored for the next iteration of the loop.

Further Expansion

No software is ever complete, so here are some ideas for additional features which could be included within this program.

Using a larger parameter table, a series of readings for each input could be stored allowing a statistical averaging routine to be used to present a smoothed average of the results over a particular number of readings. This could also incorporate a maximum and minimum storage facility which could be recalled to the screen as requested using keyboard commands.

Since some channels will not change from one reading to the next, the program could skip to the next channel if no change has occurred, thus speeding up the cycle time.

Some flicker of the bars can be seen as they are removed and replaced. Much of this could be avoided if only the relative changes were made to the length of the bar; ie shortening or lengthening it as required.

Summary

For those of you that wish to use the Hi-Res and Analogue boards in this system, this program will provide a useful tool. When setting up any control system using the Analogue board, the program will display the status of all the input channels, whatever they are monitoring.

For those of you without the necessary hardware, a careful read of the attached listing will give some idea of the power of the 6809 processor in applications such as this. ■

QL machine code programming

Michael Graham describes some of the features of the 68008, the MPU at the heart of Sinclair's QL computer.

The 68008 is just one of a family of Motorola processors and peripherals. All members of the series share the same common 32-bit architecture.

It is from this fact that Sinclair derive their claim that the QL is a 32-bit machine, the differences between the various processors being in the width of their data bus and in the sophistication of the control lines brought out from the MPU. The 68000 is in fact the most humble of the range and features an 8-bit external address bus. The advantage in employing an 8-bit bus is that the designer is able to obtain much of the performance of a 16-bit system yet make considerable savings in related hardware costs, particularly in the cost of memory.

Many of today's programmers have cut their teeth on the 8-bit micros now widely used in both home and education. While the disciplines learnt on 8-bit systems form a valuable springboard to 16-bit coding the art of producing 68008 programs will take a little getting used to. The wealth of experience of 16-bit programming resides in programmers writing for large software houses who have been producing business and applications software based on 16-bit systems for a number of years. It has already been reported that many producers of software for the home market are having difficulty in recruiting programmers with 16-bit experience despite a fairly extensive advertising effort. Those who manage to gain the necessary experience in the next few months could find themselves in a sellers market when it comes to offering their skills.

Starting point

The architecture of the 68008 has emerged from the same stable as the 6800 and 6809: MPUs that are known for the elegant simplicity of their architecture. Those with experience of these processors should have a headstart when it comes to 68008 programming. Another point of note is that the 68008 architecture is identical to the original member of the series of processors, the 68000. It was this device that formed the basis of the SAT 16 computer described in *E&CM* over recent months.

The major difference between the 68008 and 68000 is in the number of lines brought off the chip. This is obvious from the fact that the MPUs are housed respectively in a 48 and 64 pin package. One of the effects of this is that a few of the high order address lines are not brought off the chip but even so this still allows the 68008 to address over one megabyte of memory directly – a vast improvement over the capability of an 8-bit micro. Another effect of the shedding of pins is that the external data bus is only 8-bits wide. This means that the 68008 is as fast as its bigger brother when accessing byte-sized operands, however when dealing with 16-bit words the processor needs to access the words, as two successive bytes. As a result the processor's throughput is less than half that of the 68000. In general, with a typical mix of 8 and 16-bit operations, the 68008 performance is about 60% of a comparable 68000 system.

68008 registers

The processor is better endowed with CPU registers than any other 16-bit device, yet because of the elegant nature of the design there are few problems in understanding the operation. The processor features seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register. Two main stack pointers are provided, a feature familiar to 6809 programmers, with one being reserved for user programs. Of the remaining fifteen registers, eight are data and seven are address registers with the current stack pointer being designated as the eighth address register to balance things up.

The fact that the processor features a large number of registers means that it can handle functions with a large number of parameters within the processor registers. This leads to programs that are both fast and efficient as the need to continually swop parameters between the processor and external memory is avoided.

The data can be addressed in three different ways. This depends on whether the data is an 8-bit word, a 16-bit word, or a 32-bit word. However each register is general purpose and it is not necessary to designate a specialised accumulator. This emphasises the fact that the 68008 allows the programmer and not the designers of the MPU to decide which registers are used.

The seven address registers can only handle either 16-bit or 32-bit words but they have an increased range of addressing modes allowing them to be used as extra stack pointers, index registers or base address pointers. Again all of these registers operate in an identical fashion with no dedicated functions to worry about.

Addressing range

The 68008 can access over one megabyte of external memory without resorting to paging or segmentation, and like all Motorola processors has a memory mapped I/O.

Instruction set

The 68008 supports a set of 56 basic mnemonics, rather less than the 8-bit

6800, but because of the regularity of the architecture the instructions can be used with almost any register, and with almost any addressing mode or data type. This results in permutations of instructions that yield an effective instruction set of many thousand distinct operations compared to the hundred or so of the 6800.

The basic instructions of the set can be divided into the following types of operation:

Data Movement
Logic
Integer Arithmetic
Bit Manipulation
Shift and Rotate
BCD
Program Control
High-Level Language Support
System Control

The instruction set is relatively easy to remember as the programmer need only remember one mnemonic for each type of operation and then specify data size and addressing modes for both the source and destination operands.

Another advantage is that the instructions are implemented in microcode rather than random logic, so that the execution of undefined instructions does not lead to unpredictable results.

Interrupt structure

Three levels of interrupt are provided for on the 68008 by use of a 3-bit interrupt mask in the MPU's status register. As soon as an interrupt is detected the processor will issue an interrupt acknowledge signal. During this signal the hardware responsible for the interrupt can indicate that program control should be given to any one of 256 interrupt service routines or to one of three service routines corresponding to the hardware interrupt priority.

Data bus

As with other Motorola MPUs the 68008 data bus is not multiplexed. This avoids the hardware complications that arise with multiplexed buses and although in some cases this scheme requires a more expensive package for the IC, current thinking is that this system is more cost effective as it saves in the cost of support ICs and it is undoubtedly quicker.

And there's more

This brief look at the features of the 68008 has only been able to touch upon the many features of the device. We have not had space to describe the program privilege mode of operation nor to describe the various ways in which the MPU communicates with other devices within a complete system. We hope to follow up this feature with a more detailed look at some of these additional features in the near future. ■

'the 68008 can access over 1 megabyte of external memory without resorting to paging'.

Table 1. 68008 instruction set

Mnemonic	Description	Mnemonic	Description
ADBC	add decimal with extend	MOVEM	move multiple registers
ADD	add	MOVEP	move peripheral data
AND	logical and	MULS	signed multiply
ASL	arithmetic shift left	MULU	unsigned multiply
ASR	arithmetic shift right	NBCD	negate decimal with extend
B_{cc}	branch conditionally	NEG	negate
BCHQ	bit test and change	NOP	no operation
BCLR	bit test and clear	NOT	one's compliment
BRA	branch always	OR	logical or
BSET	bit test and set	PEA	push effective address
BSR	branch to subroutine	RESET	reset external deuces
BTST	bit test	ROL	rotate left without extend
CHK	check register against bounds	ROR	rotate right without extend
CLR	clear operand	ROXL	rotate left with extend
CMP	compare	ROXR	rotate right with extend
DB_{cc}	test condition, decrement & branch	RTE	return from exception
DIVS	signed divide	RTR	return and restore
DIVU	unsigned divide	RTS	return from subroutine
EOR	exclusive or	SBCD	subtract decimal with extend
EXG	exchange registers	S_{cc}	set conditional
EXT	sign extend	STOP	stop
JMP	jump	SUB	subtract
JSR	jump to subroutine	SWAP	swap data register halves
LEA	load to effective address	TAS	test and set operand
LINK	link stack	TRAP	trap
LSL	logical shift left	TRAPV	trap on overflow
LSR	logical shift right	TST	test
MOVE	move	UNLK	unlink

SCICAL'S D-LOGIC

Peter Luke tries out a Computer Assisted Learning package dealing with the fundamentals of digital logic.

The cornerstone of any study dealing with logic circuitry is a grounding in the rules that govern the action of the basic gates that go to make up any logic system. Traditionally this has been achieved by letting students loose on a collection of TTL ICs, a few LEDs and switches together with a breadboarding system. The trouble with this approach is that the majority of the time is spent learning that gates can be blown by shorting their outputs and that switches bounce unpredictably rather than just adopting clean on/off states. All valuable knowledge but hardly the object of the exercise. The advent of CAL (Computer Assisted Learning) software, and in particular the D-LOGIC package from SciCal, provides a solution to these problems.

The logical choice

D LOGIC is available for both the Spectrum and BBC micros. The version we chose for review was the cassette based BBC software. The tape loaded without any problems though the manual that accompanies the cassette provides comprehensive hints for anyone suffering from problems with loading. These include the usual advice concerning the volume setting of the recorder and a suggestion that a *TV255 command is executed before loading to ensure that none of the program's display is lost at the top of the screen.

The manual introduces the concepts of on/off binary logic and truth tables in a clear and precise manner and goes on to explore the fundamentals of boolean algebra before moving on to the text that accompanies section 1 of the program. Each of the eight sections of the program can be accessed directly from the main menu although for the beginner to get the most out of the package working through the section in order is to be recommended.

Section 1 introduces the basic logic blocks namely the AND, NAND, OR, NOR and XOR gates. The type of gate to be studied is selected from a menu and once the selection has been made the logic symbol for the gate in question is displayed together with a blank truth table. It is then possible to set the inputs to the gate to each of the four possible states, the relevant line in the truth table being entered at each stage by pressing the T key.

Section 2 of D LOGIC is a test on the knowledge gained during the previous stage and again the inputs to any of the five gates can be set to any state although this time the user must supply the value at the output. A correct answer results in an entry to the truth table displayed while an incorrect response will produce a prompt pointing out the error while making the correct entry in the truth table.

The combination of gates necessary to implement half and full adders is examined in sections 3 and 4 while section 5 provides a simulation of the action of an R-S flip-flop based on two cross coupled NAND gates.

Section 6 and 7 describe bistables and J-K flip flops while the manual puts these into the context of computer memory systems.

The final section of D-LOGIC introduces a binary counter chain that can be configured as either an up or down counter. The display produced excellent results and indicates the state of each of the gates in the chain. A timing diagram is also displayed to give an overall picture of the section of the counter. To generate this display in hardware would require the services of a

Verdict

The D-LOGIC package provides an excellent way of learning the basic principles of binary logic systems. The quality of the manual and software is to be commended. With a few hours of study it should be possible to answer all of the review questions reproduced in the manual with ease, and the understanding of basic gates will provide the springboard necessary for the understanding of far more complex systems.

SciCal also produce packages that deal with the basics of Amplification(Ampli), frequency response analysis (Fresp) and operational amplifiers (Op amp). Each of these is produced to the same high standard as D-LOGIC and form an ideal introduction to the wider area of electronics. ■

SOFTWARE REVIEWS

Adam Denning reviews a selection of the latest utility software.

Exmon

BBC Model B
Beebugsoft £10 (cassette)
£27 EPROM

Exmon is an EPROM-based machine code monitor for the BBC micro and offers the machine code programmer a lot of very useful facilities. Apart from the expected functions such as memory moving, string searching, memory editing and disassembly, this ROM also offers a full 'front-panel' display that shows registers and memory.

Any register and any (RAM) memory location can of course be altered, but perhaps the strongest feature of this program is its single stepping ability. This allows the programmer to step through his/her program one instruction at a time, running it therefore at the speed he desires, with the effects of the instructions on the 6502 registers and memory locations being clearly visible. This function is implemented extremely well, and is always the most useful – but most infrequently found – tool in a monitor.

Of course, its functions don't stop there. Programs can also be run from a specified address up to a breakpoint, the registers and so on examined, and then the code run to the next breakpoint or until it terminates. Which in the case of faulty machine code programs is almost never!

Other useful functions include the ability to send a disassembly listing to a spooled file, thus enabling it to be reassembled at a later date, and more esoteric things such as changing the paged ROM currently being looked at. One gets the impression however that Beebugsoft regard this latter point as being exceptional, which it isn't, as a simple OSBYTE call is all that is needed. Some of the functions are rather useless, such as memory verify and code relocate, but in all this is a very worthwhile product. Not quite as good as System's ADE ROM (which will be reviewed next month), but then it is half the price. **AD**

Paintbox

BBC Model B
Beebugsoft £10 (cassette)
£12 disc

Not Computer Aided Design, but a very sophisticated and versatile computer aided drawing system. Paintbox will give any amateur an insight into some of the capabilities of CAD, and uses to the full the excellent graphics of the BBC micro's Mode 2 screen.

A wide range of brushes, shapes, colours and drawing facilities are available. The primary menu displays eight colours; brush size in use; four standard brushes (circle, square, triangle and 'airbrush'); three 'elastic band' cursors which can describe any line, triangle or rectangle; a mode switch for filled shapes or outline; erase; and secondary menu.

Each form is captured by moving the cursor over it and pressing the joystick fire button – simple.

The secondary menu is used to select brush size (eight are available). There are also six functions: the first, 'variable size' is used to select unlimited brush sizes; 'Grid' enables the cursor to move in discrete steps; 'Text' allows the user to enter text from the keyboard (in any size, at any location) but I could not get it to work – see below; 'Save' saves the screen to disc or cassette; 'Infill' allows any area bounded by solid lines to be filled with colour; and 'GCOL' puts the colour into GCOL logic – giving some interesting but very unpredictable effects.

Extra facilities include stripes, solid or dotted lines, and co-ordinates for fine work.

Now a few words of warning. The erase facility only allows you to erase the whole screen – not a single line or shape drawn in error. This can be costly if you make a mistake. Secondly, you need a joystick to operate this program, and an accurate one is necessary to achieve good quality results. Thirdly, Beebug give a warning to disc users that the program won't work with certain types of Watford DFS (1.2 and below). This is true as I found to my cost; the program tended to crash at awkward moments, the text facility wouldn't work, and the screen refused to save. I am however assured by Beebug that this will not happen on Acorn DFS or Watford upgrades.

Despite these few niggling complaints, Paintbox is an excellent piece of software. It is simple to use and in no more than an hour you will be zapping through the menu, astounded at your own artistic skill! The front cover picture of this magazine was drawn using Paintbox and demonstrates well its potential – what it says about the artist's ability I will leave to your own judgement. **WO**

Oricmon

Oric 1 machine code monitor
Tansoft

Even in these days of advanced high level languages there is still the need to descend (?) to the level of assembler and even machine code to produce an acceptable performance from most micros. The ORIC-1 is no exception and ORICMON is a program to help with the development of 6502 assembly language programs. In its manual ORICMON is described as a "sophisticated machine code monitor". The use of the word "monitor" is something of a misnomer in that a machine code monitor is generally reckoned to be a small program that allows you to do basic things like examine and change memory locations. ORICMON is much more than a traditional monitor program in that it contains a disassembler and a mini assembler as well as the usual memory examine and change commands. After loading from tape you can use ORICMON to enter 6502 assembly language programs in a sort of immediate mode where each instruction is changed to machine code before you enter the next line. The form of the assembly language allowed is limited to standard 6502 mnemonics and numeric addresses, ie labels are not supported. Since the source code is translated to machine code as it is typed in there is no way of listing it, editing it or saving it on tape. This is fairly restricting if you are trying to develop any large assembly language programs but the presence of the disassembler at least means that you can examine the resulting machine code in a format very similar to the assembly language that you typed in. You can also dump the resulting machine code to tape for later use.

ORICMON is most definitely a useful piece of software for anyone engaged in writing machine code on the ORIC-1, but if you are planning anything big I would still recommend a full assembler and editor. Even then ORICMON has a useful role to play as a debugging aid.

Sprites

BBC Model B
Beebugsoft £10 (cassette)
£12 (disk)

This is a very entertaining package aimed at the beginner games writer and allows the definition and use of sprites from Basic. Sprites are user-defined graphics characters (although, they are not limited to one character square size) that can be made to move independently of other sprites and quite oblivious to what is behind or in front of them, three-dimensionally speaking.

This package from Beebugsoft is supplied on cassette or disc and comes with a 33 page manual, full of facts on how to use sprites and this software in your own games. There are two different kinds of sprites – normal ones, which only have two possible manifestations, and super sprites, which can be shown on screen in four different ways. All the driving software is in machine code, but easy access is made to Basic programs by using the integer variables A% to Z% as parameter passers and calling addreses. This is a very neat way of doing things, but of course it does tie up these variables.

Apart from a program that allows you to define the shape and colour of each manifestation of each sprite, there are also the two driver routines and seven entertaining demonstrations. Only seven sprites are allowed on screen at one time, but by using a system of 'clones', this can be made to appear as 21, which should be enough aliens for anyone!

If one has to niggle, then the only bad points are that sprites and super sprites cannot be used in the same program, and the manual, in true Beebugsoft style, is written in atrocious English. There seems to be an unnecessary preponderance of commas. No matter, the programs themselves are excellent and should provide the novice games programmer with a lot of exciting games prospects and opportunities. Definitely recommended.

BOOK REVIEWS

Harry Fairhead's monthly guide

Forth for micros

Steve Oakey
Newnes Technical Books, 1984.

This volume is another in the "'x' For Micros" series from Newnes. In this case 'x' is one of the less popular languages, Forth. I say less popular only in the sense of the number of people actually using it because Forth is a language that has attracted much attention and even devotion from its followers. The problem with any new computer language is finding a way to make the transistion from wanting to know about it to knowing about it. Once you are on the inside and understand a language it is difficult to see what all the fuss was about but from the outside a new language can be impossibly cryptic. From the outside Forth looks particularly so and even when you know a little about it everything still seems back-to-front. Steve Oakey's book is a good introduction to Forth for anyone who already knows BASIC or Pascal: Forth is not a good first language. By identifying the differences and similarities with the two best known conventional languages the reader is quickly introduced to Forth. Starting from the fundamental ideas of a stack and reverse Polish notation in chapters one and two the familiar elements of programming are built up. Chapter three introduces the "word" – the Forth equivalent of the subroutine. Chapters four and five deal with flow of control, selection and repetition. The remainder of the book deals with the various elementary data types in Forth and how to construct some of the more familiar ones such as arrays. Extended I/O including disk I/O is considered in chapter eight. Chapter ten concentrates on program development and testing a subject that is often neglected in introductory books on Forth.

Steve Oakey shows it is possible to take a structured approach to Forth and so produce readable good looking programs. What is interesting is that the compare and contrast approach used earlier in the book also serves to emphasise the structure of the language, so that by the time you reach the chapter on program design you are well prepared for most of the ideas. The final chapter explains the differences between a number of different Forth implementations.

"Forth for micros" is as mature and intelligent a guide to the language as you could ask for, and is certainly to be recommended.

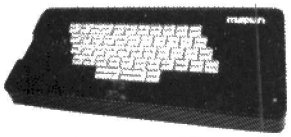
Beyond BASIC

Richard Freeman
BBC National Extension College

There comes a time in every programmer's life when need, desire or final insanity makes learning assembly language essential. The trouble is which assembly language and which machine. For, exactly like learning a high level language, there is a choice of which language and which implementation only more so.

In this book you will find not only a coherent explanation of 6502 assembly language but clarification of many of the BBC micro's MOS subroutines. Of course there are many aspects of using the BBC micro's hardware that are left out but this is not unreasonable in a book that deals with 6502 assembler in so much detail. My only criticism is that consideration of program structure is insufficiently emphasised. It is possible to write well structured programs in any language including assembler. It's all a matter of identifying the natural structuring elements – loops, decisions etc – of the language. In this sense "Beyond BASIC" is a traditional course in assembly language programming tailored for the BBC Micro and the Electron. As such it is very successful.

EXCITING ADDITIONS FOR YOUR HOME COMPUTER



KEYBOARD with ELECTRONICS for ZX SPECTRUM

★ Full size, full travel keyboard that simply plugs into expansion port on your Spectrum. ★ Offers single key selection of all major multi-key functions. ★ Extends port for other peripherals. ★ Can accept Atari-type joysticks (optional extra — order 2 of FG66W, £1.36 each and note that case will require cutting).

Three kits needed to build unit: Order LK29G, LK30H & XG35Q. Total price £39.95. Full construction details in Project Book 9 XA09K 70p. Also available ready-built. Order As XG36P. Price £44.95.

KEYBOARD with ELECTRONICS for ZX81

★ Full size, full travel keyboard that's easy to add to your ZX81. ★ No soldering in ZX81; simple instructions make it easy to fit. ★ Makes Shift Lock, Function & Graphics 2 single key selections.

Complete kit (excl. case) LW72P Price £23.95. Case XG17T £4.95. Full construction details in Project Book 3 XA03D. Price 70p. Ready-built in case XG22Y. Price £32.50.

MODEM

A CCITT standard modem that connects directly to your telephone line via a BT approved transformer. Transmits and receives simultaneously on European standard frequencies at 300 baud. May be used to talk to any other 300 baud European standard modem including the Maplin Computer Shopping modem on 0702 552941 and any British Telecom Datel 200/300 Service modem. The modem's computer interface is RS232 compatible.

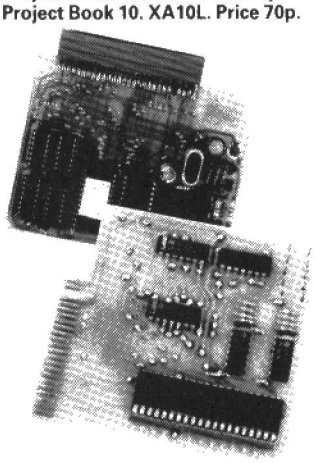
Complete kit (excl. case) LW99H. Price £44.95. Case YK62S £9.95. Full construction details in Project Book 5 XA05F Price 70p.

INTERFACES for MODEM

Interfaces are now available for the following machines: Commodore 64, Dragon 32, Oric, Spectrum, VIC20 and ZX81. Each is complete with a Machine Code Communications program. The BBC micro needs no interface and a suitable program is on Maplin catalogue page 15 or Project Book 8, page 59.

Computer	Order	Details	Price
64/VIC20	LK11M	Book 7	£9.45
Dragon 32	LK12N	Book 8	£13.75
Oric 1	LK40T	Book 10	£12.95
Spectrum	LK21X	Book 8	£17.95
ZX81	LK08J	Book 7	£24.95

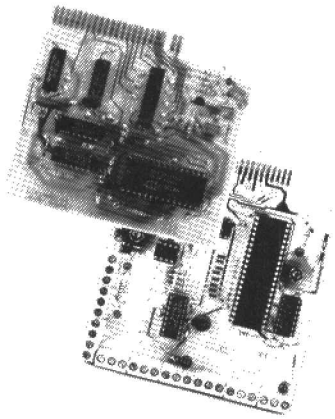
Project Book 7 XA07H. Price 70p. Project Book 8 XA08J. Price 70p. Project Book 10. XA10L. Price 70p.



ZX81 I/O PORT

★ Provides two bi-directional ports for 16 input or output lines. ★ One buffered output which can interface directly to CMOS. ★ On board address selection permits expansion to six ports with two boards.

Complete kit LW76H. Price £9.25. Full construction details in Project Book 4 XA04E. Price 70p.



MAPLIN TALK-BACK SPEECH SYNTHESISERS

★ Unlimited vocabulary with allophone (extended phoneme) system. ★ Can be used with unexpanded Oric 1, VIC20 or ZX81 as it does not require large areas of memory. ★ Speech may be easily added to programs. ★ In VIC20 version speech output is direct to TV speaker with no additional amplification needed.

Computer	Order	Details	Price
Oric 1	LK28F	Book 9	£22.95
VIC20	LK00A	Book 6	£22.95
ZX81	LK01B	Book 6	£16.95

Project Book 6 XA06G. Price 70p. Project Book 9 XA09K. Price 70p.

DRAGON 32 I/O PORT

★ Provides two TTL & 3-state bus compatible 8-bit ports, ★ Four norm/inv. latched ports, ★ Two relay switched ports ★ And two opto switched ports. ★ Module plugs directly into cartridge socket and is fully programmable from BASIC.

Complete kit LK18U. Price £13.95. Full construction details in Project Book 8 XA08J. Price 70p.

MAPLIN CATALOGUE

Full details of all Maplin's projects and electronic components in our huge 480 page catalogue. On sale now in all branches of W.H. Smith price £1.35. Or send £1.65 (incl. p&p) to our mail order address.

OTHER PROJECTS

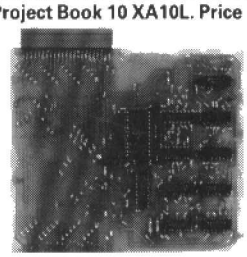
For full details of our other computer-related projects please see the relevant project book as below:

ZX81 Sound on TV — Book 6.
ZX81 Extendi-RAM — Book 9.
VIC20 Extendiboard — Book 9.
Dragon 32 Extendiport — Book 10.
TTL/RS232 Interface — Book 9.
Project Book 6 XA06G. Price 70p.
Project Book 9 XA09K. Price 70p.
Project Book 10 XA10L. Price 70p.

SPECTRUM EASYLOAD

★ Greatly reduces cassette LOADing & SAVEing problems on Spectrum. ★ Battery powered, no bus connections. ★ Charging from Spectrum PSU. ★ SAVE & LOAD indicators.

Complete kit (excl case) LK39N. Price £9.95. Full construction details in Project Book 10 XA10L. Price 70p.



ZX81 HI-RES GRAPHICS

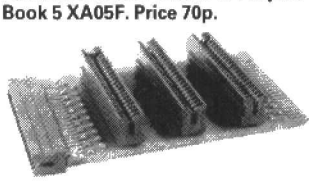
★ Full 256 x 192 fine pixel display with normal or inverted video. ★ Draws lines, circles and triangles, fills and textures. ★ Up to 32 user defined graphics. ★ Operates directly from extended BASIC.

Complete kit LK23A. Price £19.95. Full construction details in Project Book 9 XA09K. Price 70p.

ZX81 SOUNDS GENERATOR

★ Turns your ZX81 into a mini-synthesiser. ★ 3 programmable tone generators. ★ 3 programmable attenuators. ★ Noise generator with 3 pitch levels for special effect sounds. ★ Single address access with PEEK and POKE. ★ Connects directly to extension board or expansion port socket with extra socket (order RK35Q £1.96) ★ Requires separate amp and speaker.

Complete kit LW96E. Price £10.95. Full construction details in Project Book 5 XA05F. Price 70p.



ZX81 EXTENSION BOARD

★ Plugs directly into ZX81 expansion port. ★ Accepts a 16K RAM pack and three other plug-in modules simultaneously. Parts are sold separately as follows:

PCB GB08J. Price £2.40. Edge Connectors (4 needed) RK35Q. Price £1.96 each. Track pins (1 pack needed) FL82D. Price 85p per pack of 50.

MAPLIN ELECTRONIC SUPPLIES LTD

Maplin Electronic Supplies Ltd. Mail Order: P.O. Box 3, Rayleigh, Essex SS6 8LR.
Tel: Southend (0702) 552911. • Shops at: 159-161 King Street, Hammersmith, London W6. Tel: 01-748- 0926.
• 8 Oxford Road, Manchester. Tel: 061-236-0281. • Lynton Square, Perry Bar, Birmingham. Tel: 021-356-7292.
• 282-284 London Road, Westcliff-on-Sea, Essex. Tel: 0702 554000. • 46-48 Bevois Valley Road, Southampton.
Tel:0703 25831. All shops closed all day Monday.
All prices include VAT and carriage. Please add 50p handling charge to orders under £5 total (except catalogue).

WIN A ROBOT COMPETITION

your ROBOT

An EMAP Publication

BRITAINS FIRST ROBOTICS MAGAZINE

MAY 1984

TELECHIRICS - THE ART OF REMOTE CONTROL

OGRE ARM ON TRIAL

TAPPING TELEPHONES WITH BEASTY

EDITORIAL

VOLUME 1 ISSUE 4

WIN A ROBOT!

Your Robot *has arranged a competition in conjunction with the Tomy Corporation. We have two Tomy voice recognition robots and three Tomy robot arms to give away in an easy, free to enter competition.*

The voice recognition robot employs a sophisticated voice recognition circuit that understands the commands stop, talk to me, go forward, go back, turn left and turn right. It also has two arms that can respond to the spoken commands, pick up and put down.

The robot arm is driven by a single motor that via a system of clutches drives an arm that features more axes of movement than many more expensive arms. One magazine has already described the means by which it can be computer controlled and we are working on an alternative, hopefully, simpler approach.

To enter the competition simply write the answers to the questions below on a sheet of paper together with your name and address and send this to Your Robot, 2nd Floor, 155 Farringdon Road, London EC1R 3AD.

1. The Tomy voice recognition robot was recently featured in ***Your Robot's*** *news pages. In which month?*

2. Which robot made an appearance on the cover of our March issue?

3. Who manufactured the DIY robot featured as a project in the first issue of ***Your Robot****?*

4. Which company produce the Beasty servo control system?

5. Which robot featured on the front cover of the February issue of ***Your Robot****?*

The first five correct entries will receive either the Tomy Arm or voice recognition robot, and the judges' decision will be final. No correspondence in respect of this competition will be entered into and no EMAP employee or their family may enter.

Closing date will be May 31st.

CONTENTS

Editor: Gary Evans
Assistant Editor: William Owen
Contributing Editors: Gary Herman, Peter Matthews, Richard Moyle
Telephone: 01 833 0846
Advertising Manager: Richard Jansz
Telephone: 01 833 0531

YOUR ROBOT
2nd FLOOR
155 FARRINGDON ROAD, LONDON EC1R 3AD

OGRE ARM ASSESSED

William Owen makes tracks to the backwaters of Essex to examine one of the most powerful low cost robots around.

Tucked away behind an industrial estate in Barking is the small machine tool shop where the Ogre was born.

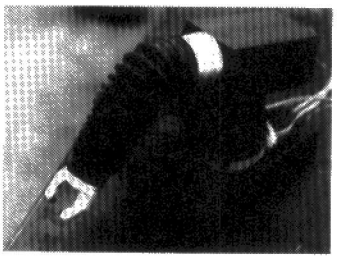
Derek and Les Staines, who run the business, used their engineering expertise (and newly acquired computer skills) to design a simple and very effective robot arm which has now gone on the market for £195.

Ogre runs on a Commodore 64 via an 8-bit controller also designed by the Staines brothers. The controller uses no relays and can simultaneously control four motors rotating in either clockwise or anti-clockwise directions. This controller (and its software) could be easily adapted to any micro with an 8-bit I/O port. It costs an extra £35 which is a competitive price in the circumstances.

NO STRINGS ATTACHED

Ogre stands about 12" high with its arm in the horizontal position. Despite its name this robot's good looks make an immediate impression. No strings, wires, pulleys or motors extrude from its rubber and polyurethane encased shell. Polyurethane mouldings are an unusual choice for a device which needs a low centre of gravity, but they are surprisingly heavy and very tough. The power units fit inside tubular Duralumin arms. The arms themselves are encased in rubber bellows and the grippers have a neat uncomplicated design. The end result is a device looking very much like a scaled down industrial robot.

Arm movement is unrestricted by the bellows. The base rotates through a full 360°, and both shoulder and elbow rotate through 60° each (about the horizontal) within a perpendicular plane. Ogre's envelope is a half sphere with a couple of inches chopped off the top.



The Ogre therefore has three axes of movement plus a powered gripper. The one obvious limitation is the omission of a rotating wrist. A fourth axis was considered by the designers but they found that the arm would become too long and lose its balance.

OPTICAL ENCODING

Direct feedback to the controller is obtained by optical encoding of the DC motors at each axis. A rotating disc with four slots is used, checking the position four times during every revolution to provide a high degree of accuracy. This works out at approximately 0.9° accuracy at the base and a little over 1° at shoulder and elbow.

The movement of each of the three encoded motors can be controlled simultaneously and movement is transmitted by worm and wheel gears which prevent 'run back' through the gear train under load.

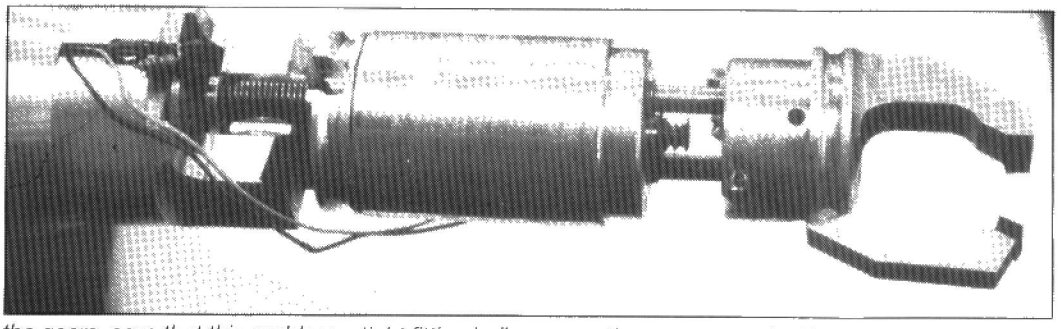
The gripper is powered by its own DC motor. It has a single moving jaw with a range of 25mm. Later models will have two jaws with a range of 50mm.

POWERLIFTING

The mythical Ogre (giant) has a reputation for strength: this

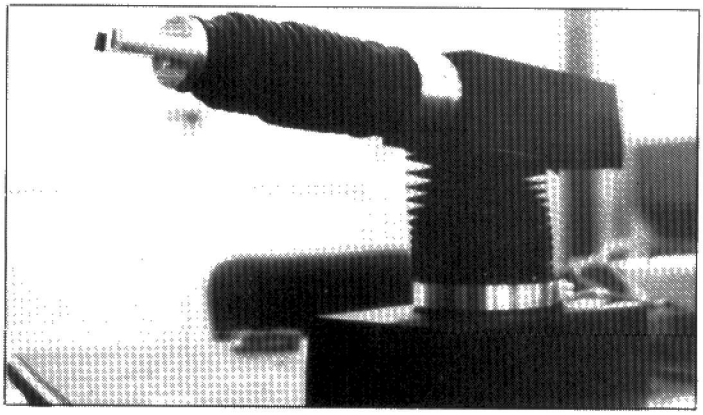
one is no exception. Derek Staines attached three Heinz soup cans to the jaws, and, if straining a little, the robot had no difficulty in making the lift. The total weight of the cans was a full 4lbs. In normal circumstances however, it is recommended that lifts of not more than 2lbs are attempted.

As in all machines strength in one area means weakness in another, and Ogre's weightlifting prowess involves a compromise with speed. To raise the arm from its lowest to highest position (a movement of some 12") took 15 seconds. The speed is doubled when both shoulder and elbow are raised simultaneously, and can be increased further by removing some of the gears. The latter course does however reduce the lifting capability.



TEETHING TROUBLES

Ogre is a young beast and could, despite its promise, do with a couple of improvements. Firstly, the prototype had a few teething troubles with its gears (excuse the pun) which meshed awkwardly making a good deal of noise. Derek Staines, who cut the gears, says that this problem will be sorted out in production models. Any improvements in this area should also quicken the arm movement. The second (small) problem is a resistor placed badly underneath the tight fitting bellows over the forearm. Movement of the bellows has a tendency to break the resistor's soldered connection; this happened during the demonstration and efforts to fix it resulted in blowing out the Commodore's keyboard! This is another problem I am assured will be rectified.



The final verdict is that Ogre will be one of the best buys around, if an when its reliability is proven. It has been designed as an expandable system. There are facilities for an extra arm, special purpose grippers, and placement on a travelling base. L. W. Staines have already received several orders, including one of 20 units from the research department of a flour mill (robots find the most unusual situations!).

The Ogre robot arm can be obtained from L. W. Staines & Co., Unit 2, Roding Trading Estate, London Road, Barking, Essex, IG11 8BU. Tel: 01-591-2900.

SOLID STATE T.V. CAMERA

Robert Harvey and Richard Sargent describe the Spectrum software that completes their five pound vision system.

The hardware of the vision system, described last month, simply consisted of a DRAM chip with its 'lid' removed. And a PIO to interface the DRAM IC to the Spectrum's data bus. From this it can be appreciated that most of the hard work is performed by the software. The software is presented in two halves. The first routines are converned with picture acquisition, where timing loops are responsible for interrogating and evaluating the capacitance (and hence the recorded light value) on a particular memory cell. The video image is obtained and placed in STORE. The later routines, specific to the Spectrum, take the information from STORE and display it on the Spectrum screen. A random element of pixel-pattern generation has been introduced in an attempt to avoid the build up of spurious patterns that can sometimes be seen when different tone values interact with each other.

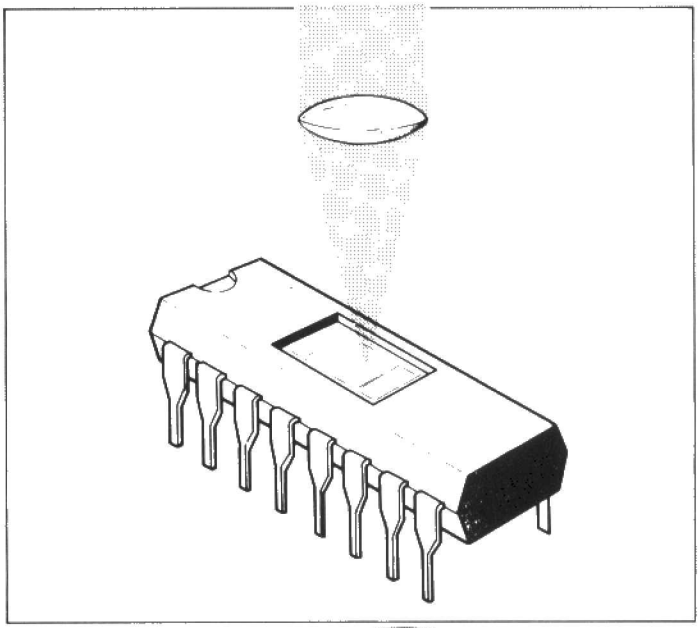
ROWS AND COLUMNS

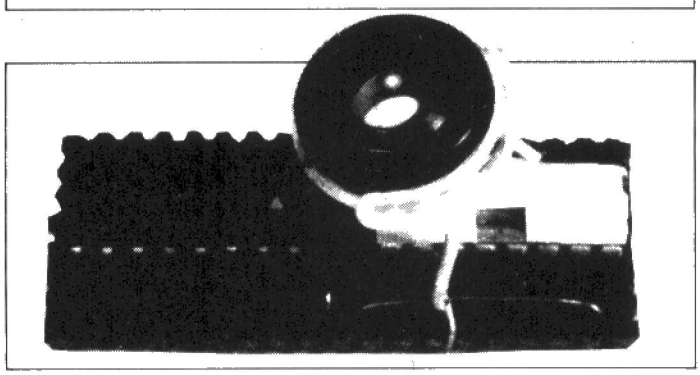
Although the rows and columns of the 128 x 128 memory matrix are perfectly laid out as far as the logic circuitry is concerned, they may not be fabricated on the chip surface in the same order as the logic implies! This will invariably result in a "scrambled" picture from the camera. Fortunately the picture is not randomly scrambled, and it can be sorted out quite easily. One method would be to instruct the software (by keypresses) to begin to re-arrange the order in which the memory cells are accessed. This method wasn't tried as it was judged (rightly or wrongly) to be a long effort in machine coding. The other method is to physically re-arrange the I/O lines going to the address pins of the 4116 until the picture quality improves. Substantial improvements in the viewed image will reveal that the wires leading to the most significant address line have been corrected, and as the less significant lines are adjusted so the picture will get clearer until the whole image is in the correct location. Changing wires on a breadboard is a speedy affair and provided the "test card" being veiwed is simple it does not take too much trial and error to obtain an accurate image.

Let's hope that robots start sprouting experimental eyes and that they see something worthwhile. ■

DRAM camera software.

```
                ;DRAM IMAGE ACQUISITION PROGRAM
                ;==============================
                ;
                ;Using 16K DRAM camera
                ;

                ;
0000            CX      EQU     0
0000            RX      EQU     0

4000            SCREEN  EQU     04000H

0001            ROWS    EQU     1       ;Rows accessed
                                        ;before delay

                ;
                        ORG     0D000H
                        LOAD    0D000H

                ;
                ;SETUP PIO - DRAM control signals
                ;
                ;Channel A:
                ;
                ;       0-6 =   A0 to A6
                ;
                ;Channel B:
                ;
                ;       0 =     Din
                ;       1 =     RAS
                ;       2 =     CAS
                ;       3 =     WE
                ;       7 =     Dout
                ;

                MAIN:
D000 ED73BDD1           LD      (BSP),SP
D004 F3                 DI

D005 01BF02             LD      BC,2BFH         ;Setup control registers
D008 3ECF               LD      A,0CFH
D00A ED79               OUT     (C),A
D00C 3E80               LD      A,80H
D00E ED79               OUT     (C),A

D010 0603               LD      B,03H
D012 3ECF               LD      A,0CFH
D014 ED79               OUT     (C),A
D016 3EF0               LD      A,0F0H
D018 ED79               OUT     (C),A

D01A 0601               LD      B,1
D01C 3E0E               LD      A,00001110B
D01E ED79               OUT     (C),A           ;Initialise cntrl lines
D020 DD21C3D1           LD      IX,CTR
                ;
                ;Picture acquisition
                ;
                ;First Clear a number of ROWS
                ;
```





The prototype vision system.

```
D024 21CAD1          LD      HL,DATA
D027 163F            LD      D,63            ;ROWS
                LP1:
D029 1E00            LD      E,0             ;COLS
D02B 22BFD1          LD      (DPTR),HL       ;Save DATA ptr
D02E ED4BC4D1        LD      BC,(BTIME)
D032 ED43C8D1        LD      (DTIME),BC
D036 ED53C1D1        LD      (ROWCOL),DE
D03A DD360001        LD      (IX+0),1        ;No of Thresholds
                LP0:
D03E D5              PUSH    DE
D03F D9              EXX
D040 D1              POP     DE
D041 0EBF            LD      C,0BFH          ;PIO address
                LP1:
D043 0600            LD      B,0
D045 7A              LD      A,D
D046 EE00            XOR     RX
D048 ED79            OUT     (C),A           ;Output RA

D04A 0601            LD      B,1
D04C ED78            IN      A,(C)
D04E F602            OR      0010B
D050 ED79            OUT     (C),A

D052 E6D5            AND     0101B           ;Drop RAS
                                             ;Drop WE
D054 ED79            OUT     (C),A
                LP2:
D056 0600            LD      B,0
D058 7E              LD      A,E
D059 EE00            XOR     CX
D05B ED79            OUT     (C),A           ;Output CA

D05D 0601            LD      B,1
D05F ED78            IN      A,(C)
D061 E60B            AND     1011B           ;Drop CAS
D063 ED79            OUT     (C),A

D065 F606            OR      0110B
D067 ED79            OUT     (C),A           ;Raise CAS
D069 1C              INC     E
D06A CB7B            BIT     7,E
D06C 20D5            JR      Z,LP1

D06E 1E00            LD      E,0
D070 15              DEC     D
D071 7A              LD      A,D
D072 E601            AND     ROWS
D074 FE01            CP      ROWS
D076 20CB            JR      NZ,LP1

D078 ED78            IN      A,(C)
D07A F60E            OR      1110B
D07C ED79            OUT     (C),A
D07E D9              EXX

                ;
                ;Delay to allow image to form
                ;

D07F CD0BD1          CALL    DELAY

                ;
                ;Now read in a number of ROWS
                ;

                LP3:
D082 D5              PUSH    DE
D083 D9              EXX
D084 D1              POP     DE
D085 D9              EXX
                LP4:
D086 0604            LD      B,4
D088 0E00            LD      C,0
                LP5:
D08A D9              EXX

D08B 0600            LD      B,0
D08D 7A              LD      A,D
D08E EE00            XOR     RX
D090 ED79            OUT     (C),A

D092 0601            LD      B,1
D094 ED78            IN      A,(C)
D096 E60D            AND     1101B
D098 ED79            OUT     (C),A

D09A 0600            LD      B,0
D09C 7B              LD      A,E
D09D EE00            XOR     CX
D09F ED79            OUT     (C),A

D0A1 0601            LD      B,1
D0A3 ED78            IN      A,(C)
D0A5 E60B            AND     1011B           ;Drop CAS
D0A7 ED79            OUT     (C),A

D0A9 ED78            IN      A,(C)           ;Get DATA bit
D0AB F606            OR      0110B           ;and raise CAS
D0AD ED79            OUT     (C),A
D0AF 1C              INC     E
D0B0 D9              EXX

D0B1 0F              RRCA
D0B2 E640            AND     40H             ;Isolate DATA
D0B4 81              ADD     A,C             ;Add on DATA
D0B5 07              RLCA                    ;Shift to next field
D0B6 07              RLCA
D0B7 4F              LD      C,A

D0B8 1C              INC     E
D0B9 10CF            DJNZ    LP5

D0BB 71              LD      (HL),C
D0BC 23              INC     HL

D0BD CB7B            BIT     7,E
D0BF 28C5            JR      Z,LP4

D0C1 1E00            LD      E,0
D0C3 15              DEC     D
```

```
D0C4 7A              LD      A,D
D0C5 E601            AND     ROWS
D0C7 FE01            CP      ROWS
D0C9 20B7            JR      NZ,LP3

D0CB E5              PUSH    HL              ;Inc DTIME
D0CC 2AC8D1          LD      HL,(DTIME)
D0CF ED4BC6D1        LD      BC,(ITIME)
D0D3 09              ADD     HL,BC           ;Next time threshold
D0D4 22C8D1          LD      (DTIME),HL
D0D7 E1              POP     HL

D0D8 D9              EXX
D0D9 ED78            IN      A,(C)
D0DB F606            OR      0110B           ;Save Power by
D0DD ED79            OUT     (C),A           ;switching off DRAM
D0DF D9              EXX

D0E0 DD3500          DEC     (IX+0)
D0E3 200A            JR      Z,LP6

D0E5 ED5BC1D1        LD      DE,(ROWCOL)     ;Reset ROWS
D0E9 2ABFD1          LD      HL,(DPTR)       ;Reset DATA ptr
D0EC C33ED0          JP      LP0
                LP6:
D0EF CD15D1          CALL    KEYS
D0F2 3E7F            LD      A,7FH
D0F4 DBFE            IN      A,(0FEH)        ;Check break keys
D0F6 E603            AND     00011B          ;Symbol shift+SPACE
D0F8 280B            JR      Z,ABORT

D0FA CB72            BIT     6,D
D0FC CA29D0          JP      Z,LP

D0FF CD50D1          CALL    DISPLAY
D102 C300D0          JP      MAIN
                ABORT:
D105 ED7B0DD1        LD      SP,(BSP)
D109 FB              EI
D10A C9              RET


                ;
                ;Delay for a while
                ;
                DELAY:
D10B ED4BC8D1        LD      BC,(DTIME)
                DEL:
D10F 0B              DEC     BC
D110 78              LD      A,B
D111 B1              OR      C
D112 20FB            JR      NZ,DEL

D114 C9              RET

                ;
                ;Allow keys to change brightness
                ;and contrast
                ;Press "b" and up/down for brightness
                ;"c" and up/down for contrast
                ;
                KEYS:
D115 E5              PUSH    HL
D116 C5              PUSH    BC

D117 3E7F            LD      A,7FH
D119 DBFE            IN      A,(0FEH)
D11B CB67            BIT     4,A     ;"B" key
D11D 2009            JR      NZ,KV10

D11F 2AC4D1          LD      HL,(BTIME)
D122 CD3CD1          CALL    UPDOWN
D125 22C4D1          LD      (BTIME),HL
                KV10:
D128 3EFE            LD      A,0FEH
D12A DBFE            IN      A,(0FEH)
D12C CB5F            BIT     3,A     ;"C" key
D12E 2009            JR      NZ,KV20

D130 2AC6D1          LD      HL,(ITIME)
D133 CD3CD1          CALL    UPDOWN
D136 22C6D1          LD      (ITIME),HL
                KV20:
D139 C1              POP     BC
D13A E1              POP     HL
D13B C9              RET

                ;
                ;Test UP/DOWN arrow keys
                ;
                UPDOWN:
D13C 011400          LD      BC,20
D13F 3EEF            LD      A,0EFH
D141 DBFE            IN      A,(0FEH)
D143 CB67            BIT     4,A
D145 2001            JR      NZ,UPD10

D147 09              ADD     HL,BC
                UPD10:
D148 CB5F            BIT     3,A
D14A 2003            JR      NZ,UPD20

D14C B7              OR      A
D14D ED42            SBC     HL,BC
                UPD20:
D14F C9              RET


                ;
                ;Convert stored data to screen display
                ;
                DISPLAY:
D150 21CAD1          LD      HL,DATA
D153 110040          LD      DE,SCREEN
D156 0E40            LD      C,64
                DS00:
D158 0620            LD      B,128/4
```

```
                DS08:
D15A C5                 PUSH    BC
D15B DD360004           LD      (IX+0),4
                DS10:
D15F AF                 XOR     A
D160 CB06               RLC     (HL)
D162 17                 RLA
D163 CB06               RLC     (HL)
D165 17                 RLA
D166 87                 ADD     A,A
D167 87                 ADD     A,A
D168 87                 ADD     A,A
D169 87                 ADD     A,A
D16A 87                 ADD     A,A
D16B 87                 ADD     A,A
D16C 4F                 LD      C,A
D16D CDADD1             CALL    RANDOM
D170 B9                 CP      C

D171 01C0C0             LD      BC,0C0C0H       ;Pattern 0
D174 3803               JR      C,DS20

D176 010000             LD      BC,00000H       ;Pattern 1
                DS20:
D179 1A                 LD      A,(DE)
D17A E63F               AND     00111111B
D17C B0                 OR      B
D17D 07                 RLCA
D17E 07                 RLCA
D17F 12                 LD      (DE),A

D180 14                 INC     D
D181 1A                 LD      A,(DE)
D182 E63F               AND     00111111B
D184 B1                 OR      C
D185 07                 RLCA
D186 07                 RLCA
D187 12                 LD      (DE),A
D188 15                 DEC     D

D189 DD3500             DEC     (IX+0)
D18C 20D1               JR      NZ,DS10

D18E 1C                 INC     E
D18F 23                 INC     HL              ;Move DATA ptr
D190 C1                 POP     BC
D191 10C7               DJNZ    DS08

                ;
                ;Move down two TV scan lines
                ;

D193 1D                 DEC     E
D194 14                 INC     D
D195 14                 INC     D
D196 7A                 LD      A,D
D197 E607               AND     7
D199 200A               JR      NZ,DS30
```

```
D19B 7B                 LD      A,E
D19C C620               ADD     A,20H
D19E 5F                 LD      E,A
D19F 3804               JR      C,DS30

D1A1 7A                 LD      A,D
D1A2 D608               SUB     08H
D1A4 57                 LD      D,A
                DS30:
D1A5 7B                 LD      A,E
D1A6 E6E0               AND     0E0H
D1A8 5F                 LD      E,A

D1A9 0D                 DEC     C
D1AA 20AC               JR      NZ,DS00

D1AC C9                 RET

                ;
                ;Pseudo random number generator
                ;
                RANDOM:
D1AD 3ABCD1             LD      A,(SEED)
D1B0 47                 LD      B,A
D1B1 17                 RLA
D1B2 A8                 XOR     B
D1B3 17                 RLA
D1B4 78                 LD      A,B
D1B5 17                 RLA
D1B6 32BCD1             LD      (SEED),A
D1B9 E67F               AND     7FH
D1BB C9                 RET


D1BC 00         SEED:   DB      0       ;Random number seed

D1BD 0000       BSP:    DW      0       ;Basic Stack Ptr
D1BF 0000       DPTR:   DW      0       ;DATA ptr
D1C1 0000       ROWCOL: DW      0       ;ROW COL store

D1C3 00         CTR:    DB      0       ;Threshold counter

D1C4 B036       BTIME:  DW      14000   ;Base threshold time
D1C6 0A00       ITIME:  DW      10      ;Increment time
D1C8 0000       DTIME:  DW      0       ;Delay time

                DATA:   DS      4096    ;Picture data store

                        END

Workarea - A480 to A745
ORG  end - E1CA
LOAD end - E000
```

PRISM'S MOVITS

Movits are a series of low cost robots soon to be distributed in this country by the Prism people. Ken Alexander previews the creatures.

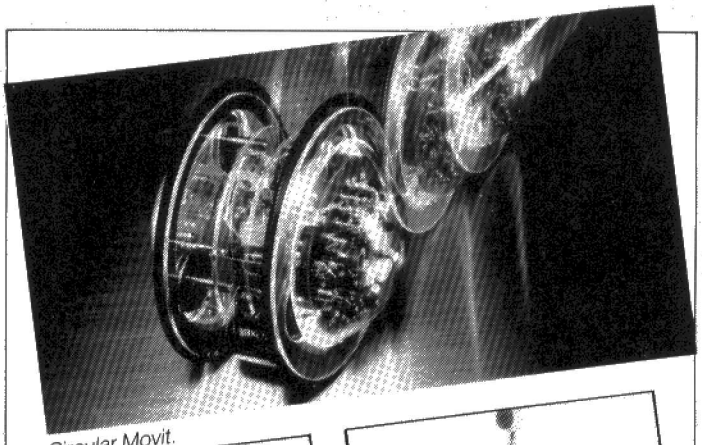
Prism Consumer Products, the people behind the £1500 Topo robot, are about to move down market with the introduction of the Movit range of robot kits. The range includes five different models ranging in price from £9.99 to £34.99. Details of the robots exact specifications were not available at the time of going to press but the sketchy details to hand indicate that the Movits will be of considerable interest to anyone experimenting with mobile robots. First apperances suggest the Movits have been designed as toys and indeed this is probably the market in which most sales will be made. The fact that they come in kit form, with detailed construction drawings would, however, tend to suggest that with a little inginuity the various members of the Movit family could be modified to operate under computer control and be used for serious experimentation.

MEETING MOVITS

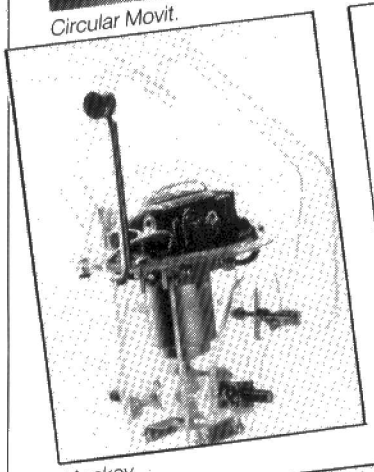
The specifications of the Movit range vary considerably, but all are motor driven from batteries mounted on board. The variations concern the sensors featured by the robots, the control systems and the method of locomotion employed.

Memocon Crawler is a basic robot that is controlled via a 5-key pad. This allows the robot to be instructed to move forward, to turn to either the right of the left, to pause, to sound an on-board buzzer or, finally, to switch on a light mounted on top of the robot. A sequence of movements may be stored in a 256 x 4K static RAM with each step of the program governing movement for between 0.7 and 0.3 seconds.

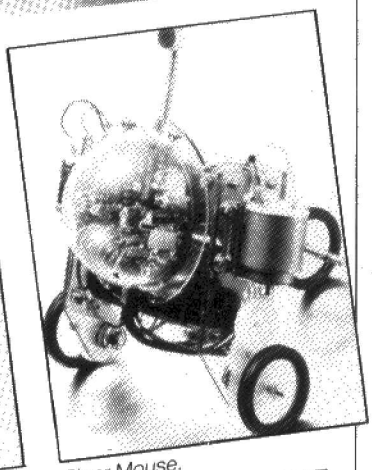
Piper Mouse is controlled via an ultrasonic controller that can be used to instruct the robot to turn to the left or right, to move forward and to stop. The futuristic design has three wheel chassis driven by two DC motors.



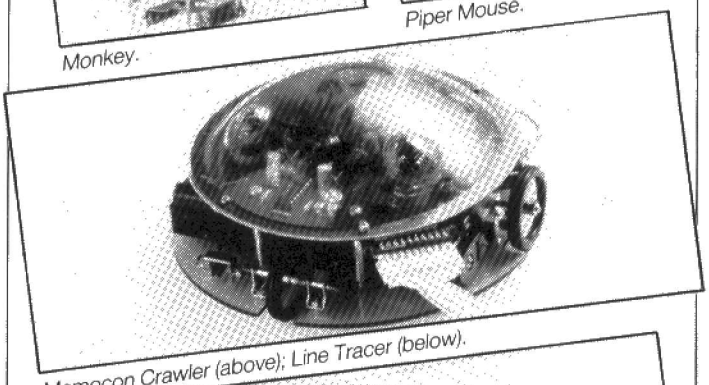
Circular Movit.



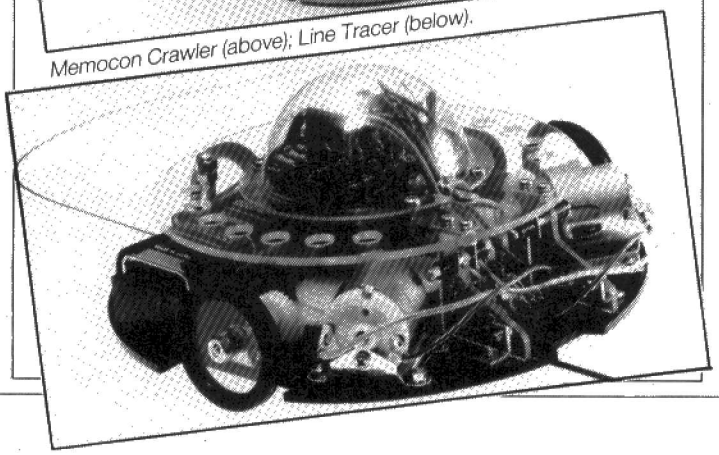
Monkey.



Piper Mouse.



Memocon Crawler (above); Line Tracer (below).



Line Tracer, as its name implies, is capable of following a black line drawn on a white background. For reliable operation the line must be at least 10mm in width and the turning radius must not be less than 15cm. This robot also features a three wheel chassis and two DC motors.

Circular returns to a fairly basic design, the robot being controlled via a cable link. The design is a departure from the norm as our photograph shows. The robot consists of two large wheels which allow the robot to be moved either in circles or in a straight line.

The last member of the family is **Monkey** and this design adopts perhaps the most original form of locomotion. Monkey hangs from a length of wire and moves along by cranking its arms back and forth. Movement is instigated by making a noise that is sufficiently loud to activate the sound sensor mounted on board.

MOVIT SURGERY

The fact that Movits are both cheap and come in kit form makes them an attractive proposition for the experimenter. The robots feature a range of senssors and control systems that if combined would result in a sophisticated device. Computer control should also be possible, particularly for Piper Mouse which has a basic ultra sonic communications link.

We will be featuring a more detailed look at Movits in the near future and, as soon as we get our hands on a couple of the variants, report on just how easy (or difficult) it is to canibalise Movits and come up with something new.

TAPPING 'PHONES WITH BEASTY

Peter Miller's novel use of a servo controller may not be the latest thing in automatic dialling, but it does present some interesting (and frivolous) possibilities.

My kitchen is piled high with technology, (no, not microwave ovens): 2 BBC computers, a modem and a Beasty instead. One object is out of place – the telephone – which must have been designed in the stone age. For £100 I could get a wizzy 50 memory auto re-dial digital telephone but isn't my kitchen too full anyway?

Last night I was reminded that it is possible to dial a number on a telephone by tapping the rest instead of rotating the dial. To dial a number one lifts the handset and taps out the number: to dial 1 tap the rest once, for 2 tap it twice and so on, and to dial 0 one must tap 10 times. After dialling each digit, leave a gap of a second or so before tapping the next.

To dial 2123 one would send the following code:

Tap-Tap – Tap – Tap-Tap – Tap-Tap-Tap

The significance of all this is that a servo is quite capable of tapping the rest, so my old telephone could be persuaded to be a 50 memory digital telephone after all.

All that is required is a BBC computer, a controller (I used Commotion's Beasty) and one standard servo.

The largest four-pronged disk that came with the servo has long arms so it can be used to press the rest directly. The servo is attached to the telephone using some thick double sided tape! (I never was a good one for details). I suggest that anyone trying this gets the software running first and sets up the servo before sticking it down.

For the sake of anyone with telephone numbers similar to the one you are dialing, do be careful to get the servo positioned correctly!

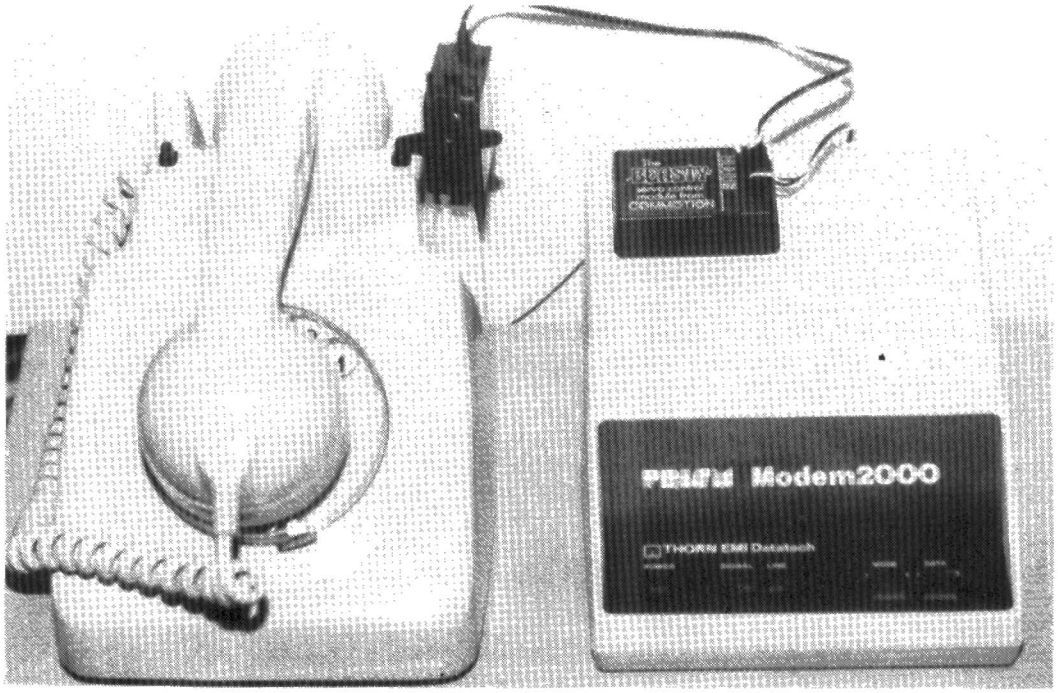
The program listed right will – **1** automatically dial a number typed at the keyboard; **2** convert a name into a telephone number and dial that; **3** auto re-dial if a blank line is entered.

```
LIST
  10  REM AUTO-DIALER
  20  REM REMEMBER TO LOAD THE BEASTY DRIVERS *LOAD DRIVER 2800
  30  DIM TABLE(26)
  40  M=190:X%=0:A$="":TABLE(0)=1
  50  DRIVER=&2800
  60  CALL DRIVER
  70  REPEAT
  80    MODE7
  90    PRINT"          AUTO-DIALLER"
 100    PRINT''
 110    RESTORE
 120    REPEAT
 130      READ NAME$,NUMBER$
 140      PRINT NAME$,,NUMBER$
 150      UNTILNAME$=""
 160    PRINT''''
 170    INPUT "Enter a number or name",NAME$
 180    IF NAME$<>"" THEN A$=NAME$
 190    RESTORE
 200    REPEAT
 210      READ NAME$,NUMBER$
 220      IF NAME$=A$ THEN A$=NUMBER$
 230      UNTIL NAME$=""
 240    IF LEN(A$)<4 OR LEN(A$)>25 PRINT"Mistake":GOTO 170
 250    I=1:J=1
 260    REPEAT
 270      IF MID$(A$,I,1)=" " OR MID$(A$,I,1)="-" THEN GOTO 310
 280      TABLE(J)=ASC(MID$(A$,I,1))-ASC"0"
 290      IF TABLE(J)=0 THEN TABLE(J)=10
 300      J=J+1
 310      I=I+1
 320      UNTIL I>LEN(A$) OR TABLE(J-1)<1 OR TABLE(J-1)>10
 330    IF TABLE(J-1)<1 OR TABLE(J-1)>10 THEN PRINT"Mistake":
        GOTO 170
 340    PRINT A$
 350    FOR DIGIT = 1 TO J-1
 360      FOR TAP = 1 TO TABLE(DIGIT)
 370        FOR DELAY = 0 TO M : NEXT
 380        Y% = 160: CALL DRIVER+3
 390        FOR DELAY = 0 TO M : NEXT
 400        Y% = 100: CALL DRIVER+3
 410        NEXT
 420      TIME = 0: REPEAT: UNTIL TIME>70
 430      NEXT
 440    UNTIL 0
 450  DATA "FRED","01234","JIM","4321","SHELA","2344"
 460  DATA "BRUCE","010 618 456 5465"
 470  DATA "",""
```

The program is divided into a number of sections

Lines 10-170	Initialisation, sets up the screen etc.
Line 180	Use the old number if a blank line was entered
Lines 190-230	Check if the input was a name given in the DATA table
Lines 240-330	Check for a valid number
Line 290	Digit 0 is in fact 10
Line 340	Print the number to be dialed
Lines 360-410	Dial the digit
Line 380	Set the servo in the pressed position
Line 400	Set the servo in the un-pressed position
Line 420	Leave a gap between digits
Lines 450-470	Dictionary of names and numbers

If only my BBC had a microphone attached to the analogue port, then it could answer the telephone when it range, and it could automatically re-dial if it heard the engaged tone, and it could be an answer-phone using a speech system to speak the message and a second servo to start a tape-recorder, then I could phone it up and use the modem to tell the computer to use a third servo to turn up the heating, and then, and then . . .



TELECHIRICS

The Greek word telechir means 'hands at a distance' and this area of robotics concerns itself with sophisticated remote control and feedback systems. Peter Matthews discusses the concept.

While a Telechir is not a robot in the conventional sense, the term, refering to remote control of systems by a human operator, its technology is certainly part of the robot revolution. The main reason for this is that it will be a few years before true robot systems will be able to imitate the skill and adaptivity of a human. Telechirics thus puts a human operator at the end of a remote control system allowing the manipulation of objects that are too dirty or dangerous to be worked upon directly. In this way the shortcomings of today's robots can be overcome. While the field of telechirics has its main applications in the industrial environment, as we shall see, there is scope for experimenting with the concepts on a home micro.

INDUSTRIAL APPLICATIONS

The main applications for both the present and near future are mainly tasks that are inaccessible to human beings, such as inside radio active establishments, sterile rooms, underwater and out in space. The other more mundane level is in dangerous work such as in factories with explosives, fire fighting and mining (what price the unmanned coal mine). Another aspect of telechirics is the ability of the controls to multiply the strength of the operator. This can take many forms. For instance a telechiric hand or manipulator could pick up a packing case of much greater weight than a human could handle by using hydraulic power. Another use is the adding of strength to artificial limbs or wasted muscles using prosthesis techniques. The use of an eckoskeleton or mechanical support framework for disabled limbs, using the signals that the brain gives to move the useless limbs is a technique already used for paralysed paraplegics. This technique is a spin-off of work done by, among others, the General Electric Corp. of America in their DEMS (Diver Equivalent Manipulator System) device. It has an electronic master arm which closely copies the human operators. It gives a proportional feedback to the operator to "feel" the necessary forces and resistance that the operating end of the arm encounters. The Slave Arm can do complex tasks, such as threading a nut on a coarse threaded bolt simply by repeating the operator's actions in doing the same thing. Sawing and the use of powered tools are easy to control, although hammering is difficult to control as the force in the arm is scaled up by a factor of 13. The slave arm is able to lift 85lbs dead weight. In addition to this the linear dimensions are scaled up by a factor of three and is hydraulically controlled.

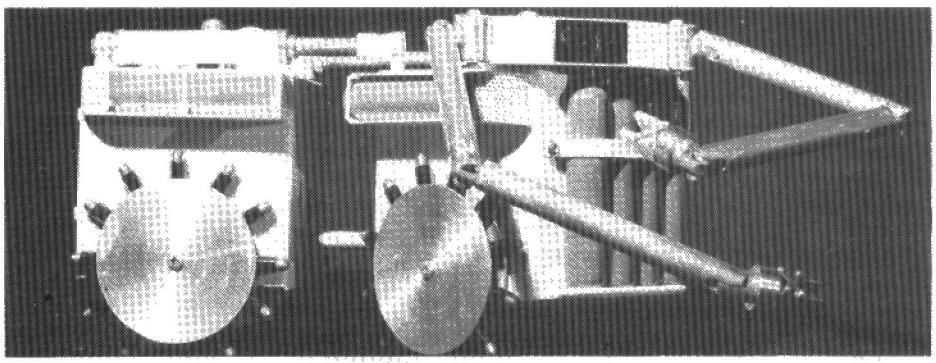
MICRO EXPERIMENTATION

Telechirics are not as remote from micro computers as might be thought. A Turtle type robot can be a kind of telechir. It is, of course possible to program it and let it carry out its tasks as a true programmed robot. It can however be controlled directly from the keyboard of a computer sending data down the umbilical cord to control and direct its wanderings.

The master arm control of a telechiric device has some similarities to a sophisticated joystick. It would be fairly simple to create a primitive telechir with a joystick, or even better a pair of them, a robot arm and a bit of software. It could be possible to build this remote controlled telechir using one of the low cost vision systems that are just beginning to come onto the market (and a long piece of ribbon cable, of course). The "State of the Art" of telechirics is not as advanced as might be possible having regard to the level of achievement of computing, low cost robots, vision and other sensors as we know them. All bilateral work in technology has been done in areas of application where there is a lot of money available ie nuclear fission, undersea work and handling in space. The term bilateral is important. It indicates a sensorary feel of the the forces and shapes that the manipulating gripper has which is reflected in the operator's controls.

This important facility of telechirics gives a feedback to the operator's master arm to give him a feel of what is happening at the slave arm.

Mans arm and hand is extremely well equipped with feedback of force sensation. Every muscle and every nerve ending is a sensor sending information back to the huge computing power of the brain which sends continuous feedback to the hand. Obviously the master arm control cannot equal human capability. What then can it do? In general a telechiric hand requires at least seven controls, three for spatial coordinates (X, Y and Z), three for spatial rotations (roll, pitch and yaw) and one for gripping. These should have direct proportional control from the movements of the operator's hand. The movement of the telechir through space (X, Y and Z) could be achieved by joystick control. The closing and opening of the gripper could be controlled by another joystick with a firing button. The master arms proportional force feed control (that is the equivalent of the feeling in your hand) is somewhat difficult to achieve however using your microcomputer. The feedback more usually comes from the sense or sighting and perception of the operator rather than feeling in the hand. If we had sophisticated force feedback which was directly proportional to the slave arms experience then we would be able to do much more complex things without looking.



Model of coal mining Telechir courtesy Professor Thring.

It would be possible to detect the weight, inertial resistance to movement and frictional or viscous force or elasticity of the object that was being held in the gripper with force feedback with such sensitive grippers it could be possible to distinguish between holding a tennis ball and a hollow steel ball of the same weight. In practice the most you might get out of a dual joystick arrangement is that you were holding an object. Even to judge how hard it was would require clever finger tip sensors.

There are of course other telechirs in the

mobile sector. These have to be attached to their computer brain by an umbilical cord if they are to qualify for the name of telechir. For this reason they are not very interesting as robot type devices (just a big turtle at heart). There are however some interesting applications such as the bomb disposal vehicle that is used by the army in Northern Ireland. Another is the vehicle developed by the German Nuclear Emergency Brigade which has a machine called EMOM. With this machine dangerous radioactive loads or spillages can be handled remotely with a hand lifting capacity of 45lbs and a load look on the elbow joint. Another mobile called the Virgula Teleoperator using a block and tackle system in the arm which gives a zero back lash to the whole complex mechanism. This has been so successful that it has been fitted to a deep sea version called ERIC 11. The first of the telechirs was developed for handling nuclear material at a distance. This is probably still the most advanced sector of the technology.

Probably the most interesting potential application of the telechir has been left to the last. The possibility of Telechiric Mining was first presented in a paper by Professor Thring of Queen Mary College way back in 1976. His thesis was that the main reason that mining for solid minerals is so much more expensive than drilling for oil or natural gas is the cost of providing a human life support below ground. The cost of ventilation, lighting, working space and transport systems for people and above all safety systems is what puts up the cost of coal. The use of telechirs at the coal face could bring down the cost of coal considerably. The fact that little or no oxygen would be needed in the mine would mean that there is no risk of combustion or explosion. There would be little need to send men down into the dark and the dust to work in one of industries most dangerous environments.

". . . Telechirics has, to date, been neglected".

In addition to this, telechiric mining could make for a wider range of mining possibilities. Mining coal could be dug at much greater depths and further out under the sea. It is only possible to ventilate a mine for human operation for 10km under the sea. A telechiric mine could operate for ten times that distance. In addition we do not mine seams less than 1 metre thick and even at this thickness the roads have to be 2 metres high. The seams could be any thickness which was practical for the machinery and the approach road need not be any thicker than the seam. A single drift would be all that was necessary as all that would be needed would be passage for the coal and the machinery. Another advantage would be that telechirics could mine seams that slope too steeply for a human miner.

The professor offers a description of a mining telechir. It consists of a mining body on four wheels which have spokes out from the rims. They grip the mine floor as it crawls forward or back and can maintain the machine on either a level or a sloping seam. The machine used in the digging mode in a confined space by the use of a series of rams. There are vertical ones in each of the two sections and others acting as a hinge between the two sections and others acting as a hinge between the two sections of body. The machine would move forward in confined spaces by operating a ram in the rear section so that the body is held rigid between the roof and the floor.

It can move forward by then extending the horizontal rams to push the front section of the body forward. This part of the body can also clamp itself between roof and floor releasing the rams in the rear section. In this mode it can advance, cutting coal at the face as it goes.

The technique of telechirics is, as we have said, been neglected. How about some of our readers developing a small telechir similar to the one that we have outlined using two joysticks (or a better system if you can think of one). We will publish the best. We may even find a disused coal mine to try it out in.